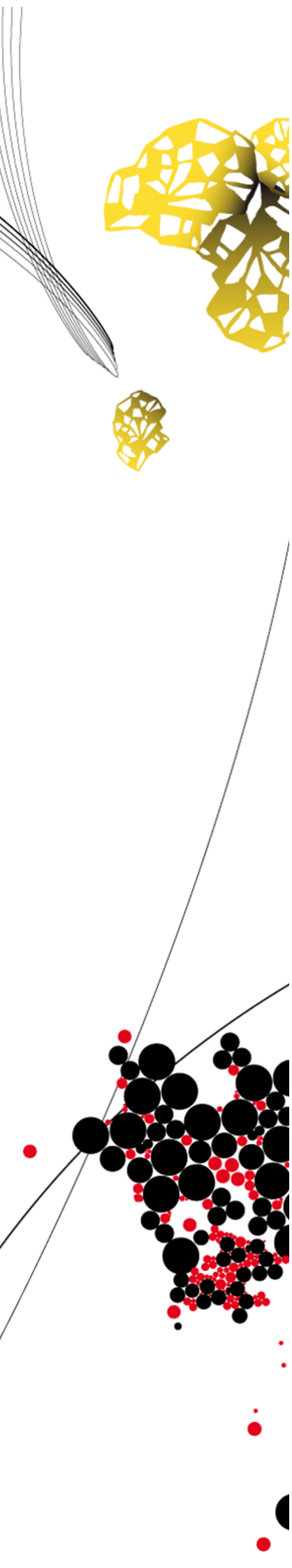

UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**

# A Deep Learning Approach to Estimating Permanents

**Brian Chang**

**B.Sc. Thesis**
**August 2018**

**Supervisors:**
dr. C.G. Zeinstra
dr. J.J. Renema

Datamanagement & Biometrics Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

# Abstract

The permanent of a matrix is a value that can be computed from a square matrix. It is calculated in almost the same way as the determinant, but all of the terms in the permanent are summed. The permanent has an application in a quantum optical experiment known as boson sampling. The probability of the outcome of this experiment can be calculated by taking the permanent of a matrix. It is believed that the complexity of approximating the outcome of this experiment is related to the photon indistinguishability. At full distinguishability, it is expected to follow polynomial complexity as the matrix size increases. At full indistinguishability, it is expected to follow exponential complexity. For partial indistinguishability, the complexity is expected to be somewhere in between. The aim of this research is to use deep learning networks to estimate the permanents for varying levels of photon indistinguishability and to investigate whether the complexity of the networks reflects the expected complexity of estimating the outcome of the boson sampling experiment.

The results of this research show a significant difference in the accuracy of estimating fully distinguishable permanents and fully indistinguishable permanents: the networks are consistently able to provide better estimates of the fully distinguishable permanent. This result appears consistent with the theory that the complexity of the fully distinguishable permanent is lower than the complexity of the fully indistinguishable permanent.

Due to the large differences in accuracy, it is not possible to make a meaningful comparison of the complexity of the networks. To try and account for the difference in accuracy, the complexity was quantified as the ratio of the number of parameters to the accuracy of the network; however, the expected polynomial and exponential complexity in the case of distinguishable and indistinguishable permanents were not observed. The results of this experiment suggest that there is some difference between the complexity of deep learning networks used to estimate distinguishable and indistinguishable permanents, but further research is needed to investigate how the complexity changes with respect to the matrix size.

# Contents

# Chapter 1

# Introduction

## 1.1 Background and Motivation

### 1.1.1 The Permanent of a Matrix

The permanent of an $N$-by-$N$ matrix $A$ with elements $a_{i,j}$ is given by the equation

$$\mathrm{Perm}(A) = \sum_{\sigma \in \Sigma} \prod_{i=1}^{N} a_{i,\sigma(i)} \tag{1.1}$$

The set $\Sigma$ is the set of all permutations of the numbers 1 to $N$. For each permutation $\sigma$, the notation $\sigma(i)$ indicates the number in the $i$-th position of $\sigma$. The sum extends over all permutations of the numbers 1 to $N$. [1] [2] For a more intuitive understanding, the permanent is simply the determinant but with all terms summed.

$$\begin{aligned} \mathrm{Det} \begin{bmatrix} a & b \\ c & d \end{bmatrix} &= ad - bc \\ \mathrm{Perm} \begin{bmatrix} a & b \\ c & d \end{bmatrix} &= ad + bc \end{aligned} \tag{1.2}$$

For something which appears so similar to the determinant on the surface, it is surprising that the computational complexity of the permanent is significantly higher. The determinant can be computed in polynomial time through Gaussian elimination, but this method cannot be used for the permanent. [3] The fastest known algorithms for calculating the permanent still have exponential complexity. In the special case where a matrix is composed entirely of real, positive elements, it is possible to approximate the permanent to a degree of error in polynomial time. [3] However, this is only an approximation, and calculating the exact value still has exponential complexity. The relevance of the permanent in the field of quantum optics, specifically the boson sampling problem, is discussed in the next section.

### 1.1.2 The Boson Sampling Problem

An interesting problem in the realm of quantum optics is the boson sampling problem. To understand this problem, it is helpful to consider an experimental implementation of boson sampling using a quantum optical network. A quantum optical network is illustrated in Figure 1.1. This network has five input modes and five output modes. In this experiment, three indistinguishable (identical) photons are injected into different modes of the network. The paths that the photons can take as they travel through the network are indicated in red. Consider the following question: what is the probability that the photons exit the network in a given combination of three unique output modes?

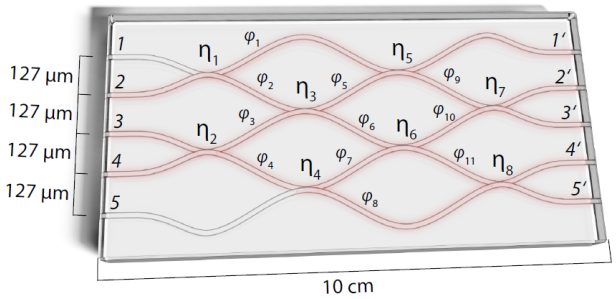


**Figure 1.1:** Illustration of a quantum optical network used for boson sampling. Image taken from [4].

It turns out that the behavior of such an optical network with $N$ inputs and $N$ outputs can be described by a unitary $N \times N$ matrix of complex numbers $U$. The probability of observing a specific combination of output modes can be calculated from a corresponding submatrix $M$ derived from $U$. [4] Consider the example optical network given in Figure 1.1 with input modes 2, 3, and 4. The input modes correspond to columns 2, 3, and 4 of the matrix $U$ and have been highlighted in red below.

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} & u_{1,5} \\ u_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} & u_{2,5} \\ u_{3,1} & u_{3,2} & u_{3,3} & u_{3,4} & u_{3,5} \\ u_{4,1} & u_{4,2} & u_{4,3} & u_{4,4} & u_{4,5} \\ u_{5,1} & u_{5,2} & u_{5,3} & u_{5,4} & u_{5,5} \end{bmatrix} \tag{1.3}$$

Suppose we are interested in the probability of observing an output in modes 1, 2, and 4. The output modes correspond to rows 1, 2, and 4 of the matrix $U$ and have

been highlighted in yellow below.

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} & u_{1,5} \\ u_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} & u_{2,5} \\ u_{3,1} & u_{3,2} & u_{3,3} & u_{3,4} & u_{3,5} \\ u_{4,1} & u_{4,2} & u_{4,3} & u_{4,4} & u_{4,5} \\ u_{5,1} & u_{5,2} & u_{5,3} & u_{5,4} & u_{5,5} \end{bmatrix} \tag{1.4}$$

The intersection of the columns corresponding to the input modes and the rows corresponding to the output modes have been indicated in orange below. These elements form the submatrix $M$ which can be used to calculate the probability of observing photons in the specified output modes.

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} & u_{1,5} \\ u_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} & u_{2,5} \\ u_{3,1} & u_{3,2} & u_{3,3} & u_{3,4} & u_{3,5} \\ u_{4,1} & u_{4,2} & u_{4,3} & u_{4,4} & u_{4,5} \\ u_{5,1} & u_{5,2} & u_{5,3} & u_{5,4} & u_{5,5} \end{bmatrix} \tag{1.5}$$

$$M = \begin{bmatrix} u_{1,2} & u_{1,3} & u_{1,4} \\ u_{2,2} & u_{2,3} & u_{2,4} \\ u_{4,2} & u_{4,3} & u_{4,4} \end{bmatrix}$$

The probability of observing photons in the specified combination of output modes is given by $|\mathrm{Perm}(M)|^2$. As discussed in the previous section, calculating the permanent has exponential complexity. Calculating this probability using a classical computer requires an exponential overhead in time and resources. This is known as the boson sampling problem. [5]

Due to the exponential increase in resources required, calculating these probabilities becomes infeasible as the size of the matrix increases. The fact that it is possible to sample from these probabilities using a quantum optical network makes it an interesting area of research for showing a quantum advantage; that is to demonstrate a quantum processing task that cannot be efficiently carried out using classical computing.

### 1.1.3 The Effect of Photon Indistinguishability

The experiment outlined in the previous section assumed that photons were fully indistinguishable (i.e. identical). In this case, the probability of observing photons in a specific combination of output modes was given by

$$|\mathrm{Perm}(M)|^2 \tag{1.6}$$

where $M$ was the corresponding submatrix derived from the unitary matrix describing the optical network $U$. However, when performing such an experiment in reality, there will always be some degree of distinguishability due to real-world limitations. [6] If the photons are not completely identical, it may be possible to differentiate photons from one another based on differences in features such as color or polarization. In the extreme case where photons are fully distinguishable, it has been shown that the probability is instead given by

$$\mathrm{Perm}(|M|^2) \tag{1.7}$$

where $M$ is the corresponding complex submatrix and $|M|^2$ denotes squared magnitude calculated element-wise. [6] This means that the permanent is now calculated from a matrix that consists of only positive, real-valued elements. For the in-between cases where there is partial photon indistinguishability, the probability is given by

$$\sum_{\sigma \in \Sigma} \left( \prod_j S_{\sigma(j),j} \right) \mathrm{Perm}(M * M^*_{1,\sigma}) \tag{1.8}$$

where $M$ is the corresponding complex submatrix, $S$ is the matrix of mutual distinguishabilities, and $\Sigma$ is the set of permutations of numbers 1 to $N$ (where $N$ is the number of rows/columns of the matrix). [6] The operator $*$ denotes the element-wise multiplication, $M^*$ denotes the element-wise complex conjugation, and $M_{1,\sigma}$ denotes that the columns of $M$ are permuted according to $\sigma$ while the rows are left unchanged. The matrix of mutual distinguishabilities is determined by properties of the photons; a more detailed treatment is beyond the scope of this research. Equation 1.8 is consistent with the previous definitions of the probability in the case of full indistinguishability (equation 1.6) and full distinguishability (equation 1.7): when the photons are fully indistinguishable, equation 1.8 reduces to equation 1.6 and when the photons are fully distinguishable, equation 1.8 reduces to equation 1.7.

It has been shown in the case of partial photon indistinguishability that the probability can be approximated to a degree of error by calculating a series of permanents of smaller matrices derived from the matrix $M$. [6] These smaller matrices are either composed entirely of complex values or real, positive values. As the indistinguishability increases, the size of the matrices of complex values increases while the size of the matrices of real, positive values decreases. At full indistinguishability, the probability is given only by a permanent with complex values (equation 1.6) and at full distinguishability, it is given only by a permanent with positive, real values (equation 1.7). [6]

Recall from section 1.1.1 that computing the permanent has exponential complexity, but approximating the permanent of a positive, real-valued matrix can be achieved with polynomial complexity for a given degree of error. However, for matri-

ces with elements that are complex numbers, this does not hold. In fact, it is conjectured that even attempting to estimate the permanent of a such a matrix results in exponential complexity (i.e. just as difficult as trying to calculate the exact value the permanent). [5] If this conjecture holds, then the complexity of approximating the permanent should depend on the degree of distinguishability. This idea forms the basis of this research. It is expected that the complexity of estimating the permanent will increase as the indistinguishability increases. At full indistinguishability, exponential complexity is expected. At full distinguishability, polynomial complexity is expected. For the partially indistinguishable cases, the complexity is expected be somewhere in between.

## 1.2 Goals of the Research

In this research, deep learning networks will be created to estimate the permanent of matrices of complex numbers. The aim of this research is to investigate whether any differences in the complexity of the networks can be observed, and whether the complexity of these networks reflects the theory. The complexity of the network is evaluated as the size of the network required to produce an estimate within a certain margin of error.

This research is divided into two parts. In the first part of the research, the computational complexity of permanents in fully distinguishable and fully indistinguishable boson sampling are considered. The permanent in the fully distinguishable case is given by $\mathrm{Perm}(|M|^2)$; estimating the permanent is expected to require polynomial complexity because the permanent is taken over a matrix consisting of only positive, real numbers. The permanent in the fully indistinguishable case is given by $|\mathrm{Perm}(M)|^2$; estimating permanent in this case is expected to have exponential complexity.

The second part of this research investigates the effect of photon indistinguishability. According to the theory, the complexity of estimating the permanent at full distinguishability should be polynomial. As the indistinguishability increases, the complexity is expected to increase as well, and at the fully indistinguishable case, the complexity is expected to become exponential. To test this theory, the best-performing networks from the first part of the research will be applied to permanents for the partially indistinguishable cases; these permanents are calculated according to equation 1.8. The complexity of approximating these permanent is predicted to increase as the photon indistinguishability increases. Due to the fact that the network size is now fixed, the increase in complexity is expected to manifest itself as a decrease in the accuracy of the networks due to the fact that the network has to approximate a more complex relationship with a limited amount of resources.

# Chapter 2

# Theory

## 2.1 Introduction to Deep Learning

Deep learning is a subfield of machine learning, which is in turn a subfield of artificial intelligence (AI). The field of artificial intelligence can be briefly summarized as “the effort to automate intellectual tasks normally performed by humans”. [7] First conceived in the 1950s, early forms of artificial intelligence followed a set of rules defined by programmers to manipulate knowledge. However, many complex problems do not follow well-defined rules, such as image classification or speech recognition. For such applications, it is impossible to program a set of explicit rules. This raises the question: instead of programming a computer with rules to process data, could a computer look at data and figure out the rules by itself?

This is the core idea behind supervised machine learning (a subfield of machine learning): a system is supplied with data as well as the expected outcome of the data, and it tries to derive the rules that relate the data to the outcome. These rules can then be applied to new data. This is the key difference between machine learning systems and early AI: machine learning systems are not programmed, they are trained. By giving the machine learning system many examples of the inputs and outputs of a task, it finds the statistical structures in these examples to come up with rules for performing the transformation from input to output. [7]

In order for a system to learn the rules of a task, three things are necessary. Firstly, the system needs example inputs of the task. Secondly, the system needs the expected output of the examples. And thirdly, the system needs some error function to measure the difference between the expected outputs and the outputs predicted by the rules it has derived so that it can evaluate its performance and make adjustments as necessary. [7]

A machine learning system tries to transform the input data into the output data. Machine learning algorithms make use of a predefined set of operations to transform the input data, such as coordinate changes or translations. [7] There are many

operations possible, some of which may result in loss of information or non-linearity. Machine learning algorithms search through this set of operations and try different transformations on the example input data. The results are compared with the expected outputs using the error function, allowing the system to evaluate whether the predictions are accurate and make adjustments to the transformations. This approach has proven to be very powerful in a solving many complex tasks.

Deep learning is subfield of machine learning and is characterized by performing many successive transformations on the input data before finally arriving at the output. Each transformation forms a layer of the model, and the number of layers is known as the depth of the model. While other approaches to machine learning tend to only have one or two layers (referred to as “shallow learning”), deep learning can involve tens or even hundreds of successive layers. [7] Deep learning networks have the potential to learn more complex relationships and may provide better results than shallow learning in more complex applications. [8]

## 2.2 Basic Overview of Neural Networks

A block diagram of a simple neural network is shown in Figure 2.1. During training, the input data is fed into the network and passed through several layers which transform the data. The transformation is controlled by weights in the layer. The last layer of the network outputs predicted values; a loss function then takes these predictions and compares them with the expected outputs to compute a loss score. The loss score gives an indication of how accurate the prediction is, and the goal is to minimize the value of the loss function. The optimizer takes the loss value and uses it to adjust the weights in the layers of the network to try and make the prediction more accurate. This cycle then repeats. [7]

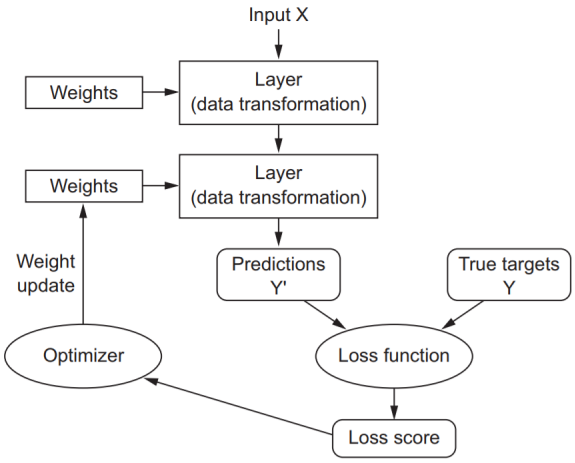


**Figure 2.1:** Block diagram of a neural network. Image taken from [7] (pg. 58).

The following sections in this chapter discuss the core concepts needed to build a deep learning network. Section 2.3 starts by considering the structure of data in a network. Section 2.4 explains how different types of layers transform input data to arrive at an output. Section 2.5 builds on this and details the process through which the network learns to adjust the parameters of the transformations. Finally, section 2.6 discusses the training and validation of a complete network.

## 2.3 Data Structure of Inputs and Outputs

### 2.3.1 Tensors

Almost all neural networks use a basic data structure known as a tensor; tensors are just generalizations of matrices to any number of dimensions. [7] Tensors of higher dimensions are created by placing tensors of lower dimensions into an array. A zero-dimensional tensor (0D tensor) is a scalar, and placing scalars into an array results in a vector or a 1D tensor. Vectors can then be placed in an array resulting in a matrix i.e. a 2D tensor. Placing matrices into an array results in a 3D tensor, and so on.

The dimensions of the input data of a network depends on the application. For example, in an network that processes grayscale images, a 2D tensor would be suitable: each element of the tensor would represent the grayscale value. For color images, each pixel can be represented as a combination of the colors red, green, and blue, so the image can be decomposed into three channels. Each RGB channel can be represented by a 2D tensor, and the channels can be placed together in an array to form a 3D tensor as illustrated in Figure 2.2.

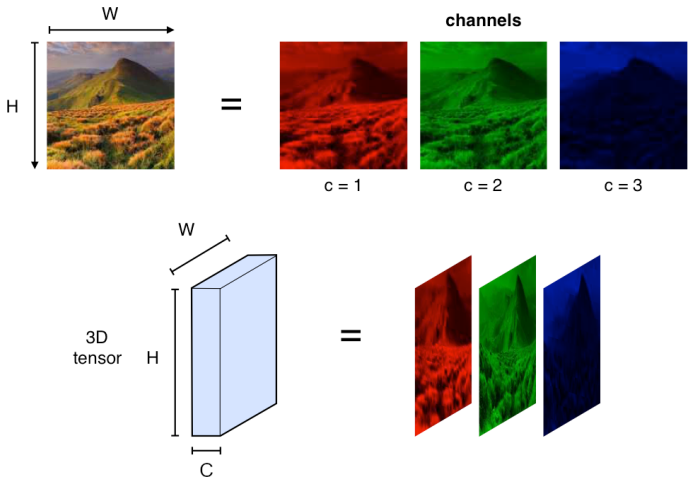


**Figure 2.2:** A color image represented as a 3D tensor. Image taken from [9].

### 2.3.2 Output Data Type

The output data type of the neural network depends on whether it is a classification model or a regression model. In short, a classification model maps the inputs to discrete output classes (also known as categories, labels, or bins), while a regression model maps the inputs to a continuous output quantity. A classification model typically outputs probabilities that the output belongs to each of the classes, and the class with the highest probability is taken as the prediction. The accuracy is calculated as the fraction of predictions which correspond to the correct classes. For a regression model, the accuracy cannot be evaluated in this manner because the output is a continuous quantity. In this case, the root mean square error is commonly used to evaluate the performance of the model (calculated as the square root of the average of the squared difference between each prediction and the true output). [10]

For the purpose of this research, a classification model will be used. This is motivated by the fact that by choosing a fixed class size, placing the permanent within the correct class corresponds to estimating the permanent within a given degree of error (namely the size of the class). An important concept in classification models is one hot encoding, which will be elaborated upon in the next section.

### 2.3.3 One Hot Encoding

One hot encoding is a process used to convert categorical data into numerical values that can be used by a network. An intuitive approach for converting classes into numerical values would be to use integers to number them, i.e. assign the first class the value “1”, the second class “2”, the third class “3”, and so on. The problem with integer numbering is that it implies certain relationships between classes. 2 is greater than 1, which implies that the second class is “greater than” the first class. The average of 1 and 3 is 2, which implies that the “average” of the first and third class is the second class. Generally speaking, these statements are incorrect or even completely nonsensical (e.g. when classifying what kind of animal is in an image). With this approach, there is the risk that the machine learning algorithm makes incorrect assumptions about the data, resulting in poor performance. One hot encoding is used to avoid this problem.

In one hot encoding, classes are encoded as vectors which have a length equal to the total number of classes. Each index of the vector corresponds to a unique class. For each encoded class, the vector will have the value 0 at all positions except for the index which corresponds to the class at which it has value 1. This is illustrated in Figure 2.3.

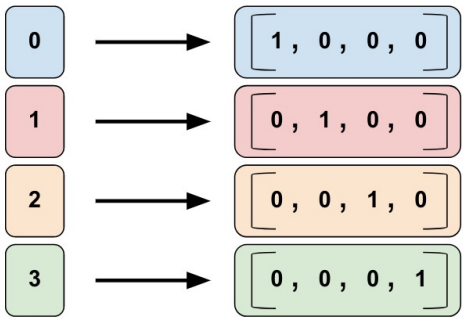


**Figure 2.3:** Illustration of one hot encoding of classes. Image taken from [11].

## 2.4 Layers and Activation Functions

### 2.4.1 A Brief Introduction to Layers

Layers are the building blocks of neural networks, and each layer provides a transformation on the input data. The simplest type of layer is a dense layer, and a small network consisting of an input layer followed by two dense layers is illustrated in Figure 2.4. The first and last layer are known as the input and output layer respectively. The layers in between are known as hidden layers; in this example, there is one hidden layer.

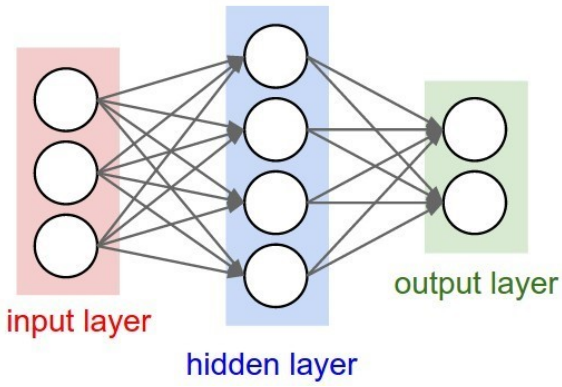


**Figure 2.4:** Diagram of a simple neural network consisting of two dense layers. Image taken from [12].

To understand the transformation performed by a dense layer, consider the second layer in this simple network. A dense layer has a number of nodes that each produce an output, and each node has a connection to every input. In this example, the dense layer consists of four nodes and each node has a connection to each of the three inputs. The operation of each node can be described the equation

$$y = f(W \bullet x) + b \tag{2.1}$$

where $x$ is the set of inputs, $y$ is the output, $W$ are the weights, $b$ is a bias value, and the function $f$ is the activation function. [7] The dot product between the inputs and weights is taken, and then an activation function is applied, after which a bias value may be added to the output. Each node has its own set of weights which is calculated when then network is trained, and a bias parameter may be specified for each node in the layer. The outputs of this layer can then be further transformed by subsequent layers.

With the understanding of how a dense layer operates, the importance of the activation function can be explained. Without an activation function, the operation performed by a dense layer would be purely linear. The output space would thus be restricted to linear transformations of the inputs, which is very limited. The purpose of the activation function is to add non-linearity to the outputs, allowing for a much richer output space. [7] Different types of activation functions are introduced in the following section.

### 2.4.2 Types of Activation Functions

There are many different types of activation functions, so this section will be limited to the ones which are relevant for this research. One of the most commonly used activation functions is the rectified linear unit, or “relu” for short. [7] The relu function is illustrated in Figure 2.5 and its operation is very straightforward: for negative values, the output is 0 and for positive values, the output is equal to the input.

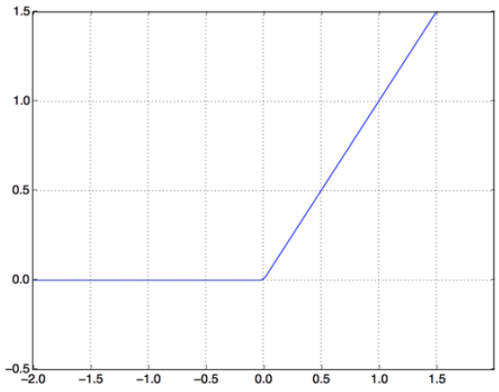


**Figure 2.5:** A plot of the relu activation function. Image taken from [7] (pg. 71).

The sigmoid activation function is illustrated in Figure 2.6 and is defined by the equation

$$y = \frac{1}{1 + e^{-x}} \tag{2.2}$$

The sigmoid functions squashes values to the range 0 to 1, so the output can be interpreted as a probability. For this reason, it is commonly used in the output layer

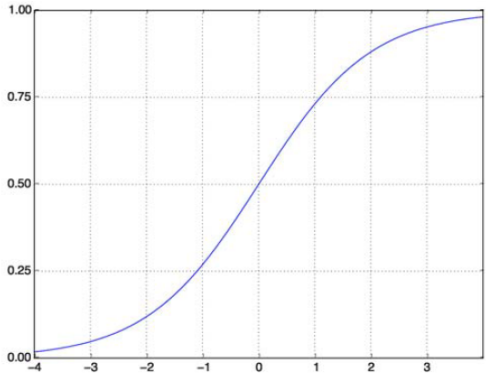


**Figure 2.6:** A plot of the sigmoid activation function. Image taken from [7] (pg. 71).

in binary classification (choosing between two classes) where it is interpreted as the probability $p$ of one of the classes. Since there are only two classes, the probability of the other class is just $1 - p$.

The softmax activation function is a generalization of the sigmoid activation function. A softmax function is described by the equation

$$y_i = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}} \tag{2.3}$$

The outputs $y_i$ scale with the inputs $x_i$, and the sum of all outputs $\sum_{i=1}^{N} y_i$ is 1. [13] Thus, it is suitable for use as an activation function in classification problems with more than two classes where the value $y_i$ is interpreted as the probability of the $i$-th class. Note that behavior of the softmax activation function is different from the aforementioned relu and sigmoid in that the softmax is dependent on the outputs of the entire layer, while the relu and sigmoid only depend on the output of the local node.

### 2.4.3 Types of Layers

There are many different types of layers than can be used in a network. As with the activation functions, this section will mostly be focused on layers which are relevant to this research. The dense layer was already discussed in section 2.1. Another type of layer is a convolutional layer. As the name suggests, a convolutional layer performs convolution operations on the data. Whereas dense layers learn global patterns in the input space, convolutional layers learn local patterns. [7] Networks that make use of convolutional layers, specifically two-dimensional (2D) convolutional layers, have shown excellent results in applications such as image classification and optical character recognition. In this research, the input data is a matrix, which also has a two-dimensional structure, so it is reasonable to try 2D convolutional layers in the network.

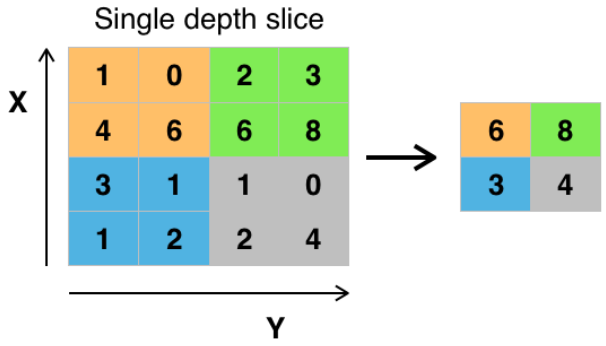


**Figure 2.7:** An illustration of a two-dimensional max pooling operation. Image taken from [14].

Another commonly used layer in neural networks for image classification is the max pooling layer. To summarize briefly, a max pooling operation effectively downsamples the input by dividing the input into regions and only preserving the maximum value in that region while discarding the rest of the values. A two-dimensional max pooling operation is illustrated in Figure 2.7. While this makes sense for image processing applications where features such as edges or textures are preserved when downsampling, the network used in this research aims to estimate the permanent of a matrix and arbitrarily discarding information is likely to result in a worse estimate. For this reason, these layers will not be used.

## 2.5 Loss Functions and Optimization

A loss function is used to evaluate the performance of the network during training so that adjustments can be made to the weights in the various layers. In a classification problem, cross-entropy loss is typically used. To consider a simple example, take a classification problem with three classes, called $A$, $B$, and $C$. The network outputs three probabilities, which correspond to the probability of each class. The cross-entropy loss is defined as the negative logarithm of the probability of the correct class. [15] Since the probability is a value which ranges from 0 to 1, the loss function is minimized when the predicted probability of the correct class is 1 (i.e. the prediction is a certainty).

The cross-entropy loss is a useful concept because it takes into account the magnitude of the probability. Suppose that the network outputs the probabilities $0.2$, $0.4$, and $0.4$ for the classes $A$, $B$, and $C$ respectively and the expected output is $A$. The class with the highest probability is $B$, so the answer is incorrect. Now suppose that after the network adjusts the weights, the probabilities are now $0.3$, $0.4$, and $0.3$. The predicted class is still incorrect, but the probability of $A$ has increased, which

means the loss has decreased. Thus, the network knows that its performance has improved to some extent, and it can continue making adjustments.

The optimizer makes use of the loss value to decide how the weights should be updated. The exact workings of optimization algorithms is beyond the scope of this research. For the purposes of this experiment, the adaptive moment estimation optimization algorithm (or "Adam") will be used. This is due to the fact that it offers good performance, has low memory requirements, and requires little tuning of its parameters to work well. [16]

## 2.6 Training and Validation

The training loop of a neural network consists of four steps: [7]

1. A batch of inputs and their corresponding expected outputs is randomly selected.

2. The inputs are run through the network and their outputs are predicted.

3. The predicted outputs are compared with the expected outputs to calculate the loss.

4. The optimizer adjusts the weights in the network to reduce the value of the loss for this particular batch.

Batches are repeatedly drawn and the training loop is repeated for each batch. An epoch is defined as a complete pass over the entire training dataset. For a batch size of 50 samples and a training set of 1,000 samples, 1 epoch takes 20 iterations. [7]

After enough iterations, the loss over the training set will become very small. However, it is important to check whether the network is learning the underlying relationship in the data or whether it is just "memorizing" the training set, known as overfitting. Random variations in training data are to be expected; for example, there could be statistical outliers in the data or measurement uncertainty. Overfitting occurs when a model fits the training data too closely, at which point it begins modeling anomalies in the data and loses sight of the underlying relationship. An overfitted model becomes too specific to the training set and will perform poorly when applied to new data.

This concept is illustrated in Figure 2.8. Two models for differentiating the blue points from the red points are illustrated here. The green line separates all of the red points from the blue points but it is overfitted: the model has become overly complex and it models all of the variations in the training set. The black line is a simpler

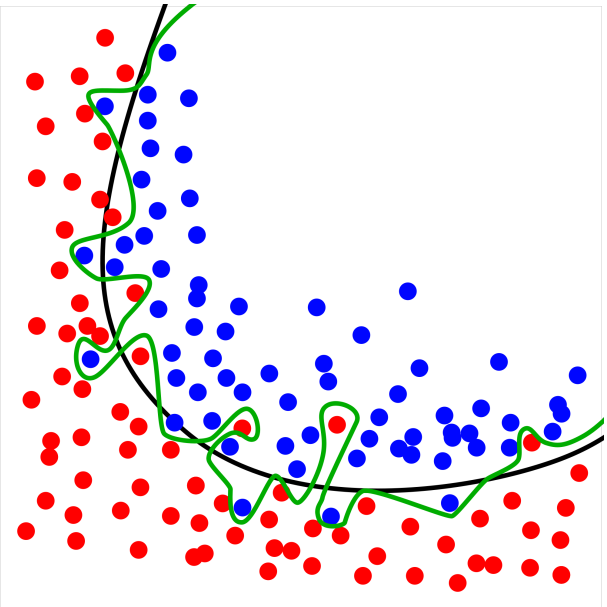

**Figure 2.8:** An illustration of a model which captures the underlying relationship in the data (black line) and an overfitted model (green line). Image taken from [17].

model that captures the underlying relationship in the data and will perform better when applied to new data.

To prevent overfitting, the dataset is generally split into two parts: a training set and a validation set. The network is trained using the training set, and the validation set is used to test the network to evaluate its performance on a dataset that has not used for training. If the loss of the validation set begins increasing (i.e. the performance is getting worse), this is an indication that the network is overfitting the training data. [7]

The converse of overfitting is underfitting, which occurs when a model is unable to capture the underlying relationship in the data. If a network is too simple compared to the underlying relationship in the data, it will be unable to accurately capture the relationship and will perform poorly on both the training and validation datasets. On the other hand, if a network is too large, it will quickly begin to overfit the training data. In deep learning (and machine learning in general), the goal is to find a balance between the two.

# Chapter 3

# Method

The deep learning networks are created in Python using the Keras framework running on the TensorFlow backend. This chapter discusses how the data is prepared and how the networks are built, trained, and evaluated.

## 3.1 Dataset

### 3.1.1 Data for Fully Distinguishable and Fully Indistinguishable Permanents

For comparing the fully distinguishable permanent with the fully indistinguishable permanent, matrices of sizes ranging from $2 \times 2$ up to $10 \times 10$ are considered. The matrices are generated using the RANDU function which generates random Gaussian-distributed numbers; Gram-Schmidt orthonormalization is then performed to produce a unitary matrix. The same matrices are used for both the distinguishable and indistinguishable case; the difference is in how the permanent is calculated. The distinguishable permanent is calculated as $\mathrm{Perm}(|M|^2)$ and the indistinguishable permanent is calculated as $|\mathrm{Perm}(M)|^2$.

The data is specified in `.TXT` files with each line corresponding to one sample. For an $N \times N$ matrix, the line contains $2N^2+2$ entries. For the first $2N^2$ entries, each pair of numbers corresponds to the real and imaginary part of an element in the matrix. For the last two entries, the first corresponds to the indistinguishable permanent for given matrix and the second corresponds to the distinguishable permanent. A function `loadData` was written to load the relevant data for training. The complete Python code is included in Appendix A.

There are 10,000 samples for each value of $N$. The matrix data is reshaped into a $10,000 \times N \times N \times 2$ array which can be interpreted as 10,000 samples of $N \times N \times 2$ arrays. The matrices are three-dimensional because there is a real and imaginary

part for each entry in the matrix.

The permanent data is loaded into a $10,000 \times 1$ array. The permanents have values which are less than 1, which makes sense because they are a probability. For classification, the logarithm of the permanent is taken because the values of the permanent are quite small ($10^{-6}$ or smaller), and this allows for the same relative error. After taking the logarithm, the permanents are separated into classes in steps of 0.1. A histogram of the 10,000 samples of the indistinguishable permanent for $N = 2$ is shown in Figure 3.1. The classes at the extreme values contain very little data, and there are numerous empty classes. To prevent this, the top 1% and bottom 1% of values are each grouped into single classes. The function `makeClasses` was written for this purpose.

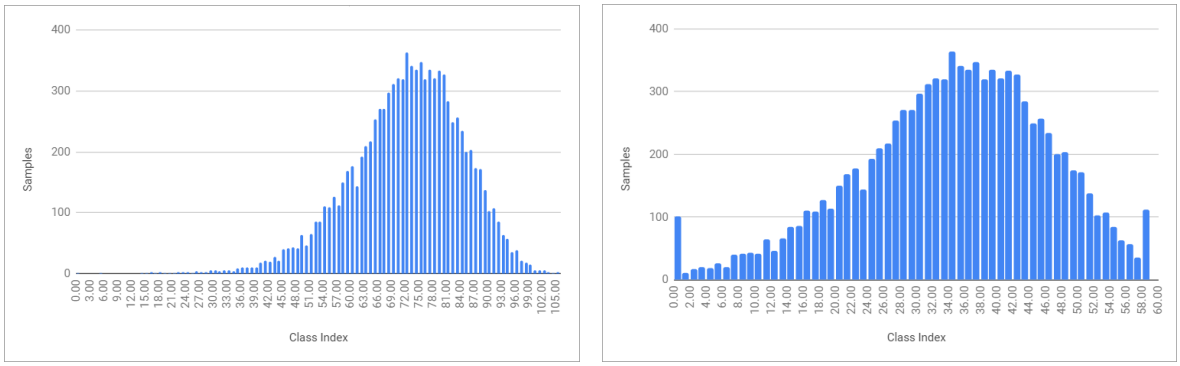


**Figure 3.1:** Histogram of 10,000 samples of indistinguishable permanents for N=2 before (left) and after (right) grouping the outlying 1% of samples.

After the classes have been made, the classes are one hot encoded using the function `to_categorical` included in the Keras framework. The datasets of matrices and their corresponding permanents are then split into two parts: 80% of the data is reserved for training and the remaining 20% is used for evaluation. The random split is made using the function `train_test_split` from the scikit-learn library. The data is now ready for training.

### 3.1.2 Data for Partially Indistinguishable Permanents

For the partially indistinguishable permanents, the data is specified in a similar format. The first $2N^2$ entries again contain the matrix data, which are random generated in the same manner as before. In addition, there are 11 additional data entries which correspond to the permanent for different levels of indistinguishability. The first is the case of 0% indistinguishability (i.e. full distinguishability), the second corresponds to 10% indistinguishability, then 20%, up to 100% (the fully indistinguishable case). These permanents are calculated according to equation 1.8 presented in

section 1.1.3. When calculating the partially indistinguishable permanent, it is assumed that each pair of photons is equally indistinguishable, that is any two photons are “as different” as any other pair.

A new function `loadInterp` was written for loading the dataset of partially indistinguishable permanents. The procedure for preparing the dataset and making classes remains the same as before.

## 3.2 Network Structure

### 3.2.1 Network Layers

The network primarily consists of 2D convolutional layers. The first layer is a 2D convolutional layer followed by a variable amount of additional 2D convolutional layers. Each 2D convolutional layer will have the same amount of filters and use the relu activation function. The amount of convolutional layers and the amount of filters in the layers will be varied during this research to find a suitable network size.

The kernel size of the convolution is $1 \times 1$. The effect of this is that the output of each convolutional layer will have the same size as before (with larger kernels it would not be possible to perform successive convolutions without padding). Convolving with a $1 \times 1$ kernel is effectively a multiplication operation but non-linearity is added by the relu activation function.

Following the convolutional layers, the data will have shape $N \times N \times C$ where $C$ is the number of filters in the convolutional layers. A `Flatten` layer reshapes the data to make it one-dimensional. The data is then passed through two dense layers. The first dense layer has a output size of 128 and uses the relu activation function. The second dense layer is the output layer, so the size is dictated by the number of classes and it uses the softmax activation function so that the output is a probability distribution that adds up to 1. The networks tested in this research will all follow this same general structure with the only variation being in the number of convolutional layers and the number of filters per layer. There will also be some variation in the output layer size due to the fact that the number of classes varies between data sets.

### 3.2.2 Loss Function and Optimizer

For classification problems, the loss function that should be used is categorical cross-entropy. It can be calculated using the equation

$$\text{loss} = -\sum_{\text{b}} \log p_n \tag{3.1}$$

where $n$ is the correct class label for each sample and $p_n$ is the predicted probability of that class. The sum is taken over all of the samples in the batch $b$. As the probabilities $p_n$ approach 1, the loss function will approach 0.

An important property of this loss function is that it only takes the probability of the correct class into account. This makes sense for a classification problem if the classes are unrelated because a prediction in another class is completely wrong. However, in this research, the classes form a numerical range, so a prediction in a class which is near the true class has a relatively small degree of error compared to a prediction which is further away from the true class. To take this into account, the loss function is modified to take into account predictions which are up to three classes away from the true prediction. The modified loss function is shown below:

$$\begin{aligned} \text{loss} = -\sum_{\text{b}} (\log(p_n) + \frac{1}{3}\log(p_{n+1}) + \frac{1}{3}\log(p_{n-1}) + \\ \frac{1}{9}\log(p_{n+2}) + \frac{1}{9}\log(p_{n-2}) + \\ \frac{1}{27}\log(p_{n+3}) + \frac{1}{27}\log(p_{n-3})) \end{aligned} \tag{3.2}$$

The weight assigned to the classes that contribute to the loss function is related to the distance from the true class. The true class is given a weight of 1, classes which are one class away are given a weight of $1/3$, then $(1/3)^2$, then $(1/3)^3$. This way, classes which are closer to the true prediction are more favored but the nearby classes are still able to contribute to the loss function. For the classes at the maximum and minimum of the range, some of these terms are unavailable so they will be omitted in the calculation.

Implementing this loss function in Keras turns out to be a complicated procedure, but fortunately there is an easier way to code this loss function. The classes have been one hot encoded, so they are vectors which have 1 in the index corresponding to the class and 0 at all other indexes. Consider an example where there are 5 classes and the second class is expected as the output. The one hot encoded vector of the second class is

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

The network outputs a vector with each element corresponding to the probability of the class. Suppose the probabilities are given by the following vector:

$$\begin{bmatrix} 0.1 & 0.5 & 0.2 & 0.1 & 0.1 \end{bmatrix} \tag{3.4}$$

In Keras, the loss function for categorical cross-entropy takes the negative logarithm of each of the probabilities, performs an element-wise multiplication with the one hot encoded vector of the expected class, and sums all of the elements. Due to the fact

that the expected class is one hot encoded, the multiplication reduces all elements except for the probability of the correct class to zero.

$$\begin{aligned} \text{loss} &= -(0 \cdot \log(0.1) + 1 \cdot \log(0.5) + 0 \cdot \log(0.2) + 0 \cdot \log(0.1) + 0 \cdot \log(0.1)) \\ &= -1 \cdot \log(0.5) \end{aligned} \tag{3.5}$$

It should now be apparent than an easy way to allow adjacent classes to contribute to the loss function is to edit the one hot encoded vector and replace their indexes with the desired weight. Suppose the one hot encoded vector is modified as follows:

$$\begin{bmatrix} \frac{1}{3} & 1 & \frac{1}{3} & 0 & 0 \end{bmatrix} \tag{3.6}$$

The loss function calculation then becomes:

$$\begin{aligned} \text{loss} &= -\left(\frac{1}{3} \cdot \log(0.1) + 1 \cdot \log(0.5) + \frac{1}{3} \cdot \log(0.2) + 0 \cdot \log(0.1) + 0 \cdot \log(0.1)\right) \\ &= -1 \cdot \log(0.5) - \frac{1}{3} \cdot \log(0.1) - \frac{1}{3} \cdot \log(0.2) \end{aligned} \tag{3.7}$$

Thus, adjacent classes can be made to contribute to the loss function by changing their values in the one hot encoded vector. Their values in the vector will correspond to their weight in the loss function. Using this approach, the loss function described in equation 3.2 is implemented. The function `fuzzEncode` is written to convert a standard one hot encoding to the modified one hot encoding.

As discussed in section 2.5, the Adam optimizer will be used. The parameters of the Adam optimizer are left unchanged; the default settings provided by Keras are used.

## 3.3 Training and Validation Process

### 3.3.1 Training of Networks for Fully Distinguishable and Fully Indistinguishable Permanents

As discussed in the section 3.2.1, the network's first layer is a 2D convolutional layer followed by a series of additional 2D convolutional layers. The number of additional layers will be varied from 1 to 6. All of the 2D convolutional layers will use the same amount of filters, which will be varied from 2 to 128 in powers of 2 (i.e. 2, 4, 8, etc.). Each of these networks will be trained for values of $N$ ranging from 2 to 10 for fully distinguishable as well as fully indistinguishable permanents. As discussed in section 3.1.1, the network will be trained on 8,000 samples and validated on 2,000 samples. A batch size of 100 is chosen for training. The Adam optimizer is used,

and the loss function is the modified form of categorical cross-entropy described in section 3.2.2.

To prevent overfitting, the validation loss will be monitored during training. After each epoch, the network will test the validation data to see if the loss on the validation dataset has improved. If the loss does not improve for 10 consecutive epochs, training will cease. This is done using the Keras `EarlyStopping` callback and setting a patience value of 10. The networks are trained until the validation loss stops improving.

Apart from the loss and categorical accuracy, several other metrics are tracked during training. The predicted class is the class with the highest probability, and the categorical accuracy is defined as the number of correct predictions divided by the total number of predictions. In addition to this, a function `cat_acc_margin` has been defined to calculate the number of correct predictions within a certain margin of error, with the margin being the number of classes between the prediction and the true value. The accuracy within a margin of 1 to 10 classes is monitored. Additionally, the top-5 accuracy is also tracked. This is defined as the fraction of times that the true class is among the five classes with the highest probabilities. These metrics are calculated every epoch and saved.

At the end of training, all of the aforementioned metrics are written to `.TXT` files and the model is saved. In addition to this, the network is tested on the entire training set and validation set and all of the metrics are calculated and saved. The number of classes, number of epochs, training time, and number of parameters in the network are also recorded. The function `saveResults` is written for this purpose.

### 3.3.2 Training of Networks for Partially Indistinguishable Permanents

For the investigation of partially indistinguishable permanents, the best-performing network structure for each $N$ from the first part of the research will be used. Using the first part of the research, the number of layers and filters that yields the best results at each $N$ is determined. A network with this structure is then used to train networks on each of the partially indistinguishable cases. The training and validation process remains the same as in the first part of the research.

# Chapter 4

# Results

## 4.1 Comparison of Fully Distinguishable and Fully Indistinguishable Permanents

The deep learning networks have the amount of 2D convolutional layers varied from 1 to 6 (not counting the input layer). For each of these amounts of layers, the number of filters is varied from 2 to 128 in powers of 2. This leads to 42 possible combinations of layers and filters. Each of these network configurations is trained on both the distinguishable and indistinguishable permanent for a matrix of size $N$ with $N$ ranging from 2 to 10. This leads to a total of 756 unique networks.

For the distinguishable permanent, a summary of the three networks that produced the highest validation categorical accuracy ("Acc ($\pm$0)") for each $N$ is presented in Table 4.1 (source data included in Appendix B). "L" is the number of additional convolution layers, "C" is the number of filters in each of the convolutional layers, "Parameters" is the number of trainable parameters (weights) in the network, "Acc ($\pm$0)" is categorical accuracy, "Acc ($\pm$3)" is accuracy within 3 classes, "Acc ($\pm$10)" is accuracy within 10 classes, and "Acc (Top-5)" is how often the correct class is within the top five predictions. This summary has also been compiled for the indistinguishable permanent in Table 4.2 (source data in Appendix C).

*Table 4.1: Highest Accuracy Networks for Distinguishable Permanent.*

| N | L | C | Parameters | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|---|
| 2 | 4 | 32 | 28,443 | 0.603 | 1.000 | 1.000 | 0.995 |
| 2 | 4 | 128 | 139,707 | 0.570 | 1.000 | 1.000 | 0.999 |
| 2 | 4 | 64 | 57,339 | 0.549 | 0.999 | 1.000 | 0.993 |
| 3 | 4 | 128 | 221,240 | 0.348 | 0.982 | 1.000 | 0.962 |
| 3 | 6 | 32 | 50,648 | 0.336 | 0.974 | 0.999 | 0.939 |

*Continued on next page*

*Table 4.1 – continued from previous page*

| N | L | C | Parameters | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|---|
| 3 | 4 | 64 | 97,912 | 0.330 | 0.972 | 1.000 | 0.940 |
| 4 | 5 | 16 | 41,528 | 0.196 | 0.860 | 0.997 | 0.747 |
| 4 | 5 | 32 | 78,264 | 0.186 | 0.831 | 0.995 | 0.721 |
| 4 | 6 | 8 | 24,192 | 0.182 | 0.835 | 0.997 | 0.708 |
| 5 | 5 | 16 | 59,444 | 0.177 | 0.788 | 0.997 | 0.669 |
| 5 | 4 | 8 | 32,748 | 0.170 | 0.803 | 0.998 | 0.684 |
| 5 | 6 | 8 | 32,892 | 0.168 | 0.826 | 0.999 | 0.690 |
| 6 | 5 | 16 | 81,972 | 0.138 | 0.765 | 0.998 | 0.614 |
| 6 | 5 | 4 | 25,380 | 0.126 | 0.700 | 0.995 | 0.558 |
| 6 | 4 | 4 | 25,360 | 0.121 | 0.687 | 0.994 | 0.550 |
| 7 | 3 | 16 | 107,923 | 0.129 | 0.593 | 0.986 | 0.493 |
| 7 | 1 | 16 | 107,379 | 0.122 | 0.591 | 0.981 | 0.468 |
| 7 | 2 | 32 | 209,619 | 0.121 | 0.601 | 0.983 | 0.466 |
| 8 | 5 | 8 | 72,498 | 0.135 | 0.642 | 0.991 | 0.501 |
| 8 | 6 | 128 | 1,154,610 | 0.129 | 0.636 | 0.989 | 0.492 |
| 8 | 4 | 8 | 72,426 | 0.127 | 0.620 | 0.986 | 0.500 |
| 9 | 6 | 16 | 174,146 | 0.116 | 0.604 | 0.984 | 0.468 |
| 9 | 6 | 8 | 89,978 | 0.116 | 0.632 | 0.995 | 0.517 |
| 9 | 6 | 32 | 344,786 | 0.113 | 0.636 | 0.990 | 0.502 |
| 10 | 2 | 4 | 57,959 | 0.119 | 0.628 | 0.989 | 0.493 |
| 10 | 4 | 32 | 420,627 | 0.118 | 0.629 | 0.989 | 0.479 |
| 10 | 6 | 32 | 422,739 | 0.116 | 0.640 | 0.993 | 0.503 |

*Table 4.2: Highest Accuracy Networks for Indistinguishable Permanent.*

| N | L | C | Parameters | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|---|
| 2 | 5 | 64 | 63,176 | 0.108 | 0.517 | 0.834 | 0.428 |
| 2 | 2 | 64 | 50,696 | 0.108 | 0.516 | 0.823 | 0.426 |
| 2 | 5 | 32 | 31,176 | 0.105 | 0.526 | 0.842 | 0.433 |
| 3 | 5 | 16 | 30,030 | 0.052 | 0.262 | 0.677 | 0.209 |
| 3 | 4 | 32 | 51,374 | 0.049 | 0.254 | 0.671 | 0.197 |
| 3 | 3 | 4 | 14,870 | 0.049 | 0.279 | 0.693 | 0.208 |
| 4 | 2 | 4 | 18,950 | 0.044 | 0.231 | 0.594 | 0.169 |
| 4 | 4 | 16 | 44,610 | 0.042 | 0.249 | 0.616 | 0.170 |
| 4 | 5 | 8 | 27,474 | 0.042 | 0.236 | 0.590 | 0.189 |

*Continued on next page*

*Table 4.2 – continued from previous page*

| N | L | C | Parameters | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|---|
| 5 | 5 | 32 | 118,611 | 0.038 | 0.191 | 0.538 | 0.161 |
| 5 | 4 | 8 | 36,747 | 0.038 | 0.211 | 0.556 | 0.146 |
| 5 | 1 | 32 | 114,387 | 0.038 | 0.186 | 0.534 | 0.151 |
| 6 | 6 | 8 | 48,284 | 0.035 | 0.193 | 0.516 | 0.148 |
| 6 | 5 | 128 | 683,732 | 0.033 | 0.177 | 0.484 | 0.125 |
| 6 | 6 | 32 | 164,852 | 0.033 | 0.186 | 0.513 | 0.137 |
| 7 | 2 | 2 | 23,655 | 0.033 | 0.171 | 0.465 | 0.126 |
| 7 | 1 | 32 | 212,949 | 0.032 | 0.171 | 0.470 | 0.139 |
| 7 | 4 | 128 | 880,341 | 0.032 | 0.169 | 0.459 | 0.119 |
| 8 | 2 | 2 | 27,366 | 0.029 | 0.164 | 0.451 | 0.109 |
| 8 | 5 | 4 | 43,844 | 0.029 | 0.161 | 0.441 | 0.120 |
| 8 | 2 | 64 | 543,764 | 0.028 | 0.172 | 0.463 | 0.116 |
| 9 | 6 | 4 | 53,084 | 0.033 | 0.167 | 0.457 | 0.110 |
| 9 | 6 | 32 | 349,688 | 0.027 | 0.164 | 0.444 | 0.110 |
| 9 | 4 | 8 | 94,736 | 0.027 | 0.145 | 0.412 | 0.104 |
| 10 | 1 | 64 | 834,774 | 0.028 | 0.152 | 0.403 | 0.109 |
| 10 | 3 | 32 | 424,086 | 0.027 | 0.152 | 0.428 | 0.116 |
| 10 | 3 | 128 | 1,699,542 | 0.026 | 0.148 | 0.429 | 0.115 |

Across the board, the accuracy of the best-performing networks for estimating the distinguishable permanent is significantly higher than the best-performing networks for the indistinguishable case. The highest categorical accuracy achieved in the indistinguishable case is just 10.8% for $N = 2$, which is significantly lower than the distinguishable case which achieved an accuracy of 60.3%. In both cases, the categorical accuracy decreases for larger values of $N$, as illustrated in Figure 4.1.

A plot of the highest top-5 accuracy achieved at each $N$ is shown in Figure 4.2. Again, the distinguishable permanent has significantly higher accuracy that the indistinguishable case. The categorical accuracy shown in Figure 4.1 tends to be lower than the top-5 accuracy which is expected.

The categorical accuracy within a margin of 3 classes is plotted in Figure 4.3. The classes have a width of 0.1 and the scale is logarithmic, so predictions which are 3 classes away from the true class are off by a factor of 2. The accuracy within a margin of 3 classes is higher than the top-5 predictions. This indicates that in some cases, the highest predicted class is not too far from the true class despite the fact that the true class is not among the top-5 predictions.

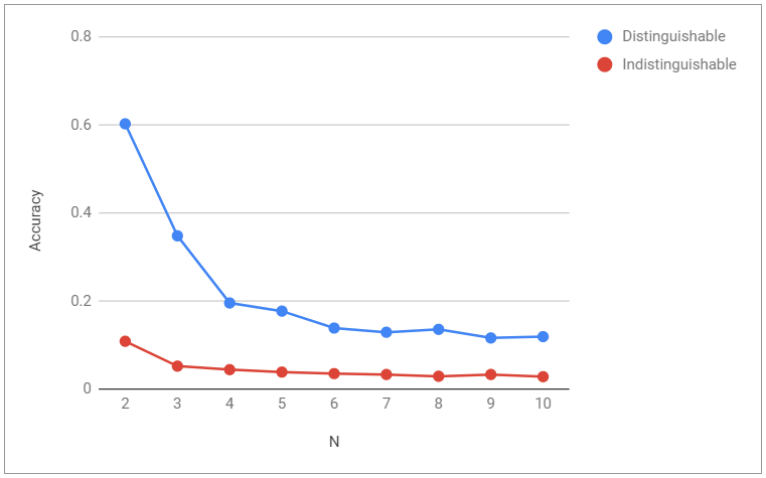


**Figure 4.1:** Validation categorical accuracy for the distinguishable and indistinguishable permanent.

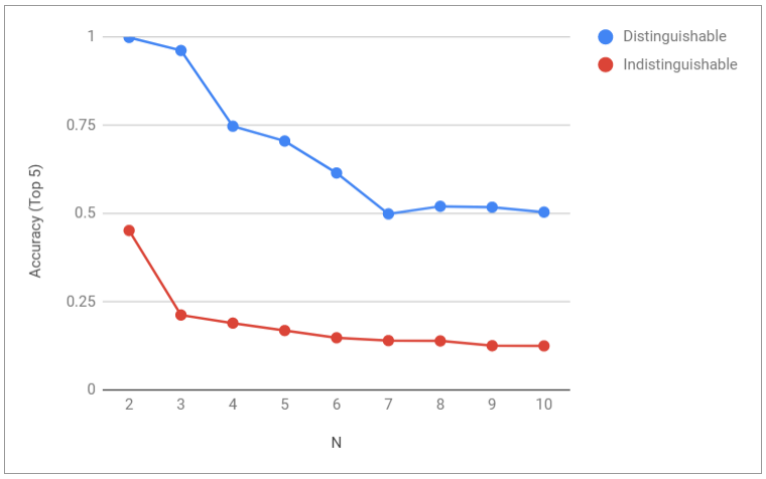


**Figure 4.2:** Validation top-5 accuracy.

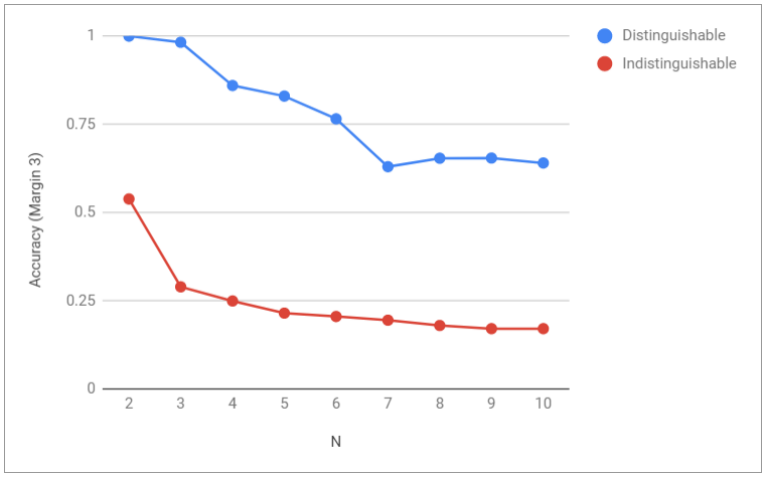


**Figure 4.3:** Validation categorical accuracy within a margin 3 classes.

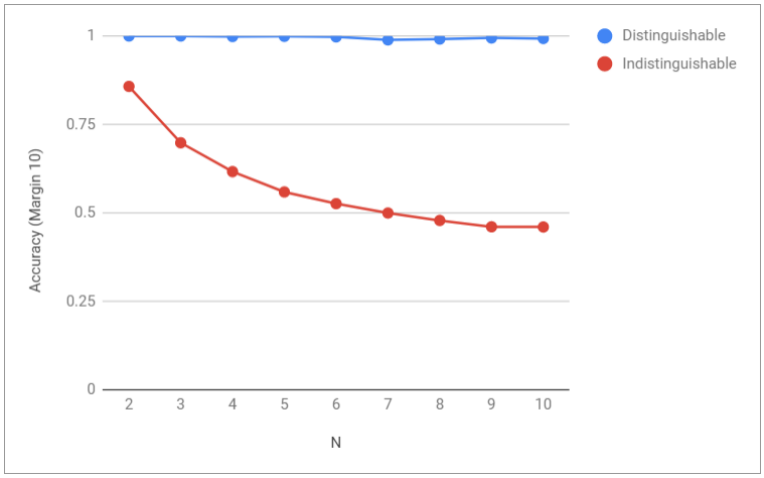


**Figure 4.4:** Validation categorical accuracy within a margin of 10 classes.

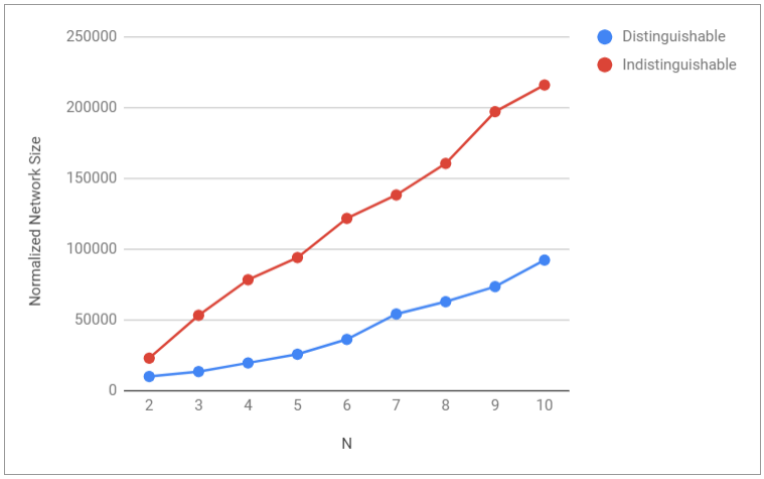


**Figure 4.5:** Normalized network size.

The categorical accuracy within a margin of 10 classes is plotted in Figure 4.4. To put this into perspective, predictions which are 10 classes away are off by a factor of 10. For the distinguishable case, the networks are consistently able to achieve about 99% accuracy in this regard, so that their predictions are at least in the same order of magnitude as the expected results. For the indistinguishable permanents, the decrease in accuracy for higher $N$ is still observed.

The number of parameters in the network gives an indication of the size of the network and it will divided by the accuracy to take into account the performance of the network. The ratio of parameters divided by accuracy will be referred to as the normalized size of the network.

The networks with the poorest accuracy are for $N = 10$ in the indistinguishable case, which can only achieve an accuracy within a margin of 3 classes of about 17%. Thus, when considering the size of the network, 16% is taken as the minimum

accuracy that the network must reach to be considered. For each $N$, the normalized size of each tested network is calculated to find the minimum, and the results have been plotted in Figure 4.5.

## 4.2 Investigation of Permanents in the Case of Partial Indistinguishability

For investigating the effect of partial indistinguishability, the networks with the best results in the first part of this research will be used. At each $N$, the combination of layers and filters that lead to the highest accuracy will be used. Since the accuracy of the networks for indistinguishable permanents tends to be lower than the distinguishable permanent, the best-performing networks for the indistinguishable permanent will be chosen. The validation categorical accuracy within a margin of 3 classes is used because the categorical accuracy with no margin is generally too small to make an accurate comparison. A summary of the number of layers and filters in the best-performing networks for $N$ ranging from 3 to 7 is given in Table 4.3.

*Table 4.3: Number of Layers and Filters in Highest Accuracy Networks.*

| N | Layers (L) | Filters (C) | Acc ($\pm$3) |
|---|---|---|---|
| 3 | 3 | 8 | 0.289 |
| 4 | 4 | 16 | 0.249 |
| 5 | 5 | 64 | 0.214 |
| 6 | 6 | 64 | 0.205 |
| 7 | 5 | 16 | 0.195 |

The combination of layers and filters that yields the best result for each $N$ will be used, e.g. for $N = 3$, a network with 3 additional layers and 8 filters will be trained for each of the partially indistinguishable cases. The indistinguishability is defined as an index ranging from 0 to 1 where 0 is fully distinguishable and 1 is fully indistinguishable. Steps of 0.1 are taken; for each step there is a unique set of permanents which is used to train the network.

A plot of the accuracy achieved by each of the networks at the different levels of indistinguishability is shown in Figure 4.6 (source data included in Appendix D). For each of the values of $N$, the accuracy decreases as the indistinguishability increases. Interestingly, the accuracy observed for $N = 2$ was lower than the other values of $N$.

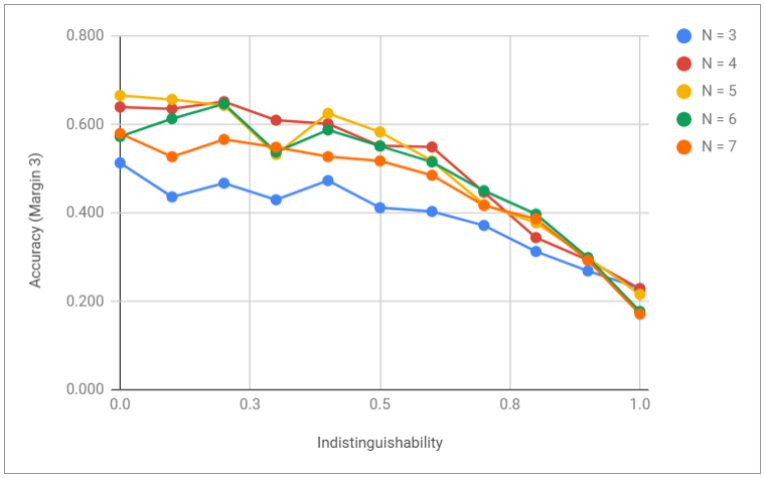


**Figure 4.6:** Accuracy at different levels of indistinguishability for fixed network sizes.

# Chapter 5

# Discussion

In the first part of this research, networks of varying sizes were tested on fully distinguishable permanents and fully indistinguishable permanents. The aim was to investigate whether a difference in the complexity of the networks could be observed, and whether it reflects the theory. The number of additional convolutional layers was varied from 1 to 6 and the number of filters was varied from 2 to 128 in powers of 2 to try and find a network which could produce a good approximation.

The results of the experiment showed a significant difference in the highest validation accuracy that could be achieved for the distinguishable and the indistinguishable permanents. The accuracy that could be achieved for the distinguishable permanent was considerably higher. In both cases, the accuracy decreased for larger network sizes.

The number of parameters in the network is taken as an indicator of the complexity of the network, but due to the fact that there is a large difference in the accuracy of the networks for the distinguishable and indistinguishable case, performing a direct comparison is not fair. To try and account for the difference in accuracy, the number of parameters is divided by the accuracy (margin 3) to normalize the network size. This way, both the accuracy as well as the complexity required to achieve this accuracy are taken into account. Networks with an accuracy below 16% are not considered because 16% is the highest common accuracy that could be achieved at all $N$ for both the distinguishable and indistinguishable case. This is done to help filter out networks which may have very small sizes but poor predictive power (ideally this threshold would be set higher but this was the highest common accuracy that could be achieved in this research). The networks for the indistinguishable permanent had larger normalized sizes than the networks for distinguishable permanents, and in both cases the normalized size increased for larger $N$. It was not possible to observe exponential/polynomial growth in the normalized size with respect to $N$.

It should be noted that the networks for larger $N$ will be somewhat larger due to the fact that there are more inputs. Furthermore, the networks for the indistinguish-

able permanents are somewhat larger than the networks indistinguishable permanents due to the fact that the indistinguishable permanents have a wider distribution of values, resulting in more classes and hence more parameters. The fact that there are more classes is also likely to lower the accuracy to some extent. However, these factors alone are not enough to account for the large difference in the accuracy of the networks for the distinguishable and indistinguishable permanents.

In the partially indistinguishable case, the accuracy observed for $N = 2$ was lower than the other values of $N$. Generally speaking, higher accuracy is achieved at lower values of $N$. This discrepancy may be explained by the fact that the networks used in the partially indistinguishable cases have a fixed structure which is based on the fully indistinguishable case, so they may not provide optimal results when used to train on permanents of lower indistinguishability.

# Chapter 6

# Conclusion and Recommendations

## 6.1 Conclusion

The results of this research suggest that there is some difference between approximating the distinguishable and indistinguishable permanent of a matrix. The networks for approximating the distinguishable permanents consistently achieved higher accuracy compared to the indistinguishable permanents. Further research needs to carried out to investigate whether the difference in the accuracy is a result of a difference in the complexity of approximating these permanents.

The nature of the relationship between the size of the matrix and the complexity of estimating the permanent with a deep learning network remains uncertain. Due to the large difference in the accuracy that the networks were able to achieve, the complexity was estimated by dividing the number of parameters in the network by the accuracy to produce a normalized size. It was not possible to observe exponential/polynomial growth in the normalized size with respect to $N$.

The results also suggest that the indistinguishability may be related to the complexity of approximating the permanent. At a fixed network size, the networks for approximating the permanent consistently perform better when the indistinguishability is lower. Again, more research needs to be carried out to verify whether this difference in accuracy is a result of the difference in complexity.

## 6.2 Recommendations

In the first part of this research, many different networks were tested on varying values of $N$ for distinguishable and indistinguishable permanents. In order to allow for systematic testing, the main part of the network only used one type of layer (2D convolution) and the parameters that were varied were the number of layers and number of filters. For future research, a good starting point would be to test

networks consisting of more types of layers.

Even for small $N$, there was a considerable difference between the accuracy of the networks for the distinguishable and indistinguishable permanents. It should be investigated whether better estimates of the indistinguishable permanents can be made. It only makes sense to compare the complexity of the networks for estimating indistinguishable permanents with the networks for estimating distinguishable permanents if they are able to achieve similar levels of accuracy.

There may also be possibilities to improve the dataset used for training. In the future, larger training and validation datasets can be considered. The datasets can also be normalized so that the distribution of samples is even across the classes. This may allow for better results when training. It could also be investigated whether other data representations may allow for better results: for example, the complex numbers are currently specified as a real and imaginary part, but they can also be represented in modulus-argument form.

# Bibliography


[1] Wikipedia, “Permanent (mathematics),” 2018, last accessed 8 August 2018. [Online]. Available: https://en.wikipedia.org/wiki/Permanent_(mathematics)

[2] ——, “Determinant,” 2018, last accessed 8 August 2018. [Online]. Available: https://en.wikipedia.org/wiki/Determinant

[3] ——, “Computing the permanent,” 2018, last accessed 8 August 2018. [Online]. Available: https://en.wikipedia.org/wiki/Computing_the_permanent

[4] M. Tillmann, B. Daki, R. Heilmann, S. Nolte, A. Szameit, and P. Walther, “Experimental boson sampling,” vol. 7, December 2012.

[5] B. T. Gard, K. R. Motes, J. P. Olson, P. P. Rohde, and J. P. Dowling, “An introduction to boson-sampling,” 27 June 2014.

[6] J. J. Renema, A. Menssen, W. R. Clements, G. Triginer, W. S. Kolthammer, and I. A. Walmsley, “Efficient algorithm for boson sampling with partially distinguishable photons,” 10 July 2017.

[7] F. Chollet, *Deep Learning with Python*. Manning Publications Co., 2018.

[8] E. Dezhic, “What is deep learning and its advantages,” 26 June 2017, last accessed 12 August 2018. [Online]. Available: https://becominghuman.ai/what-is-deep-learning-and-its-advantages-16b74bc541a1

[9] A. Bursuc, F. Krzakala, and M. Lelarge, “Lecture 6: Neural networks, convolutions, architectures,” 2017, last accessed 12 August 2018. [Online]. Available: https://becominghuman.ai/what-is-deep-learning-and-its-advantages-16b74bc541a1

[10] J. Brownlee, “Difference between classification and regression in machine learning,” 11 December 2017, last accessed 12 August 2018. [Online]. Available: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/

[11] TensorFlow, “Feature columns,” 8 August 2018, last accessed 12 August 2018. [Online]. Available: https://www.tensorflow.org/guide/feature_columns

[12] A. Sukhadeve, “Understanding neural network: A beginners guide,” 6 August 2017, last accessed 12 August 2018. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/understanding-neural-network-a-beginner-s-guide

[13] E. Murhabazi, “What ive learned from my udacity deep learning course (sigmoid function vs softmax function),” 20 October 2017, last accessed 12 August 2018. [Online]. Available: https://medium.com/@EspyMur/what-ive-learned-from-my-udacity-deep-learning-course-sigmoid-function-vs-softmax-function-e3e122334f34

[14] Wikipedia, “Convolutional neural network,” 2018, last accessed 13 August 2018. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network

[15] J. D. McCaffrey, “Why you should use cross-entropy error instead of classification error or mean squared error for neural network classifier training,” 5 November 2013, last accessed 13 August 2018. [Online]. Available: https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/

[16] Chengwei, “Quick notes on how to choose optimizer in keras,” February 2017, last accessed 13 August 2018. [Online]. Available: https://www.dlology.com/blog/quick-notes-on-how-to-choose-optimizer-in-keras/

[17] Wikipedia, “Overfitting,” 2018, last accessed 13 August 2018. [Online]. Available: https://en.wikipedia.org/wiki/Overfitting

# Appendix A

# Python Code

```
# Import libraries
# Used for various calculations
import numpy as np
# Used loading data
import pandas as pd
# Used for splitting data into training and validation sets
from sklearn.model_selection import train_test_split
# Used for one hot encoding
from keras.utils import np_utils
# Various Keras things
import keras
from keras.models import Sequential
from keras import layers
import keras.backend as K
import tensorflow as tf
# Used for measuring run time
import time

# Choose indistinguishable (hard) and/or distinguishable (easy)
easy = False
hard = False

# Choose partially indistinguishable permanents
interp = True
interp_steps = np.round(np.arange(0.0,1.1,0.1), decimals=1)

# Make evenly sized classes based on distribution of data (don't use)
even_distribution = False
```

```
# Use the same number of classes for the indistinguishable permanent
# Requires "easy = True"
copy_easy_cats = False

# Matrix size start/end
N_start = 3
N_stop = 3

# Layers start/end
l_start = 3
l_stop = 3

# Filters start/range (incremented in powers of 2, up to a maximum of
# 2**(c_range-1))
c_start = 2
c_range = 8

# Trials start/end
trial_start = 1
trial_stop = 1

# Number of data samples
nsamp = 10000

# Upper and lower percentile of data to group
outlierpercentile = 1

#Ratio of the data to use for validation (between 0 and 1)
vdata = 0.2

# Batch size, maximum epochs, and patience for training
batch = 100
max_epoch = 1000
patience = 10

# Modify one hot encoding to use adjacent classes
# spread = number of adjacent classes on each side
# factor = factor to divide by each step
```

```
fuzzEncode = True
spread = 3
factor = 3

# Initialize number of classes
# Overwritten automatically if not using even_distribution
nclasses = 50
save_nclasses_easy = np.zeros((N_stop - N_start + 1, trial_stop -
                               trial_start + 1), 'int')

# Load function for partially indistinguishable permanent
# Sources used:
# [1] https://www.datacamp.com/community/tutorials/deep-learning-
#     python
def loadInterp(N, samples, hardness):
    # Generate string for path to data
    path = "".join(("D:\\Documents\\Module 12\\Data\\interp", str(N),
                    ".txt"))
    # Import data
    dataset = pd.read_csv(path, sep=" ", header=None)
    # Convert X part of DataFrame to array
    X = dataset.iloc[0:samples, 0:N*N*2].values
    # Choose the output data
    y = dataset.iloc[0:samples, N*N*2 + int(10*hardness)].values
    return X, y

# Load function for distinguishable/indistinguishable permanent
# Sources used:
# [1] https://www.datacamp.com/community/tutorials/deep-learning-
#     python
def loadData(N, samples, hard):
    # Generate string for path to data
    path = "".join(("D:\\Documents\\Module 12\\Data\\mlbs", str(N),
                    ".txt"))
    # Import data
    dataset = pd.read_csv(path, sep=" ", header=None)
    # Convert X part of DataFrame to array
    X = dataset.iloc[0:samples, 0:N*N*2].values
    # Choose the output data
```

```
    if hard:
        y = dataset.iloc[0:samples, N*N*2].values
    else:
        y = dataset.iloc[0:samples, N*N*2 + 1].values
    return X, y

# Create classes
def makeClasses(y, percentile):
    # Prepare output
    out = np.log(y)
    # Make a sorted array to determine class bounds
    ysort = np.sort(out)
    samples = np.shape(y)[0]
    # Determine indexes of upper and lower percentile
    lbound = int(percentile / 100 * samples) - 1
    ubound = int((100 - percentile) / 100 * samples)
    # Get values at upper and lower percentile
    lbound = ysort[lbound]
    ubound = ysort[ubound]
    # Convert into integer representation for processing
    lbound = int(lbound*10)
    ubound = int(ubound*10) - 1
    # Begin classification.
    ncats = ubound - lbound + 2
    for val in np.nditer(out, op_flags=['readwrite']):
        for cat in range(ncats):
            if (-lbound - cat) + (val*10) < 0:
                val[...] = cat
                break
            if cat == ncats - 1:
                val[...] = cat
    return ncats, out

# Make a specified number of classes with a fixed width
def cloneClasses(y, percentile, ncats):
    # Prepare output
    out = np.log(y)
    # Make a sorted array to determine class bounds
    ysort = np.sort(out)
```

```
    samples = np.shape(y)[0]
    # Determine indexes of upper and lower percentile
    lbound = int(percentile / 100 * samples) - 1
    ubound = int((100 - percentile) / 100 * samples)
    # Get values at upper and lower percentile
    lbound = ysort[lbound]
    ubound = ysort[ubound]
    # Convert into integer representation for processing
    lbound = int(lbound*10)
    ubound = int(ubound*10) - 1
    # Begin classification.
    for val in np.nditer(out, op_flags=['readwrite']):
        for cat in range(ncats):
            if ((-lbound - cat * ((ubound - lbound + 2) / ncats)) +
                (val*10) < 0):
                val[...] = cat
                break
            if cat == ncats - 1:
                val[...] = cat
    return out

# Make variable-width classes that have the same numebr of samples
# Bad results, don't use.
# Sources used:
# [1] https://docs.scipy.org/doc/numpy-1.14.0/reference/generated/
#     numpy.arange.html
# [2] https://docs.scipy.org/doc/numpy-1.14.0/reference/generated/
#     numpy.stack.html
# [3] https://stackoverflow.com/questions/2828059/sorting-arrays-in-
#     numpy-by-column
def distributionBasedClass(y, classes):
    # Prepare output
    out = np.log(y)
    # Make a sorted array to use as indexes
    samples = np.shape(y)[0]
    srcindex = np.arange(samples)
    # Add indexes as a second column
    out = np.stack((out, srcindex), axis=1)
    # Sort on size of first column
```

```
    out[out[:,0].argsort()]
    ## Classify
    for cat in range(classes):
        start = int(cat * samples / classes)
        stop = int((cat + 1) * samples / classes)
        out[start:stop,0] = cat
    # Revert order
    out[out[:,1].argsort()]
    return out[:,0]

# Add extra terms to OHE for loss function
def fuzzOHE(data, spread, factor):
    for row in range(data.shape[0]):
        for col in range(data.shape[1]):
            if data[row, col] != 0:
                for blah in range(-spread, spread+1):
                    if col + blah >= 0 and col + blah < data.shape[1]:
                        asdf = factor**-abs(blah)
                        data[row, col + blah] = asdf
                break

# Modified categorical accuracy to allow margin of error in adjacent
# classes
# [1] https://github.com/keras-team/keras/blob/master/keras/metrics.py
# [2] https://www.tensorflow.org/versions/r1.1/api_docs/python/tf/
#     logical_or
def cat_acc_margin(y_true, y_pred, margin):
    # Default categorical accuracy
    acc = K.equal(K.argmax(y_true, axis=-1), K.argmax(y_pred, axis=-1))
    # Take logical "or" with adjacent classes
    for i in range(margin + 1):
        acc = tf.logical_or(acc, K.equal(K.argmax(y_true, axis=-1),
                                         K.argmax(y_pred, axis=-1) - i))
        acc = tf.logical_or(acc, K.equal(K.argmax(y_true, axis=-1),
                                         K.argmax(y_pred, axis=-1) + i))
    return K.cast(acc, K.floatx())

# Calls to modfied categorical accuracy
def cat_acc_m1(y_true, y_pred):
```

```
    return cat_acc_margin(y_true, y_pred, 1)

def cat_acc_m2(y_true, y_pred):
    return cat_acc_margin(y_true, y_pred, 2)

def cat_acc_m3(y_true, y_pred):
    return cat_acc_margin(y_true, y_pred, 3)

def cat_acc_m4(y_true, y_pred):
    return cat_acc_margin(y_true, y_pred, 4)

def cat_acc_m5(y_true, y_pred):
    return cat_acc_margin(y_true, y_pred, 5)

def cat_acc_m6(y_true, y_pred):
    return cat_acc_margin(y_true, y_pred, 6)

def cat_acc_m7(y_true, y_pred):
    return cat_acc_margin(y_true, y_pred, 7)

def cat_acc_m8(y_true, y_pred):
    return cat_acc_margin(y_true, y_pred, 8)

def cat_acc_m9(y_true, y_pred):
    return cat_acc_margin(y_true, y_pred, 9)

def cat_acc_m10(y_true, y_pred):
    return cat_acc_margin(y_true, y_pred, 10)

# Log training statistics
class logTraining(keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.loss = []
        self.cat_m0 = []
        self.cat_m1 = []
        self.cat_m2 = []
        self.cat_m3 = []
        self.cat_m4 = []
        self.cat_m5 = []
```

```
        self.cat_m6 = []
        self.cat_m7 = []
        self.cat_m8 = []
        self.cat_m9 = []
        self.cat_m10 = []
        self.cat_top5 = []

    def on_epoch_end(self, epoch, logs={}):
        self.loss.append(logs.get('loss'))
        self.cat_m0.append(logs.get('categorical_accuracy'))
        self.cat_m1.append(logs.get('cat_acc_m1'))
        self.cat_m2.append(logs.get('cat_acc_m2'))
        self.cat_m3.append(logs.get('cat_acc_m3'))
        self.cat_m4.append(logs.get('cat_acc_m4'))
        self.cat_m5.append(logs.get('cat_acc_m5'))
        self.cat_m6.append(logs.get('cat_acc_m6'))
        self.cat_m7.append(logs.get('cat_acc_m7'))
        self.cat_m8.append(logs.get('cat_acc_m8'))
        self.cat_m9.append(logs.get('cat_acc_m9'))
        self.cat_m10.append(logs.get('cat_acc_m10'))
        self.cat_top5.append(logs.get('top_k_categorical_accuracy'))

# Log validation statistics
class logTesting(keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.loss = []
        self.cat_m0 = []
        self.cat_m1 = []
        self.cat_m2 = []
        self.cat_m3 = []
        self.cat_m4 = []
        self.cat_m5 = []
        self.cat_m6 = []
        self.cat_m7 = []
        self.cat_m8 = []
        self.cat_m9 = []
        self.cat_m10 = []
        self.cat_top5 = []
```

```
    def on_epoch_end(self, epoch, logs={}):
        self.loss.append(logs.get('val_loss'))
        self.cat_m0.append(logs.get('val_categorical_accuracy'))
        self.cat_m1.append(logs.get('val_cat_acc_m1'))
        self.cat_m2.append(logs.get('val_cat_acc_m2'))
        self.cat_m3.append(logs.get('val_cat_acc_m3'))
        self.cat_m4.append(logs.get('val_cat_acc_m4'))
        self.cat_m5.append(logs.get('val_cat_acc_m5'))
        self.cat_m6.append(logs.get('val_cat_acc_m6'))
        self.cat_m7.append(logs.get('val_cat_acc_m7'))
        self.cat_m8.append(logs.get('val_cat_acc_m8'))
        self.cat_m9.append(logs.get('val_cat_acc_m9'))
        self.cat_m10.append(logs.get('val_cat_acc_m10'))
        self.cat_top5.append(logs.get('val_top_k_categorical_accuracy'))

# Save results and model
# Sources used:
# [1] https://stackoverflow.com/questions/45199047/how-to-save-model-
#     summary-to-file-in-keras
from contextlib import redirect_stdout
def saveResults(N, nlayers, csize, ncats, batch, max_epoch, patience,
                X_train, y_trainOHE, X_test, y_testOHE, trial, hard,
                ctype, model, logTrain, logTest, training_time,
                interp, hardness):
    #Setup path and labels
    folder = "D:\\Documents\\Module 12\\Test\\"
    labels = ["Loss", "Acc_0", "Acc_1", "Acc_2", "Acc_3", "Acc_4",
              "Acc_5", "Acc_6", "Acc_7", "Acc_8", "Acc_9", "Acc_10",
              "Acc_T5"]

    #Set variables
    L = nlayers
    C = csize
    T = trial

    if interp:
        Type = "".join(("Interp", str(hardness)))
    elif hard:
        Type = "Hard"
```

```
else:
    Type = "Easy"

if ctype == 1:
    classtype = "Even Distribution"
elif ctype == 2:
    classtype = "Clone Easy"
elif ctype == 3:
    classtype = "Fixed Step"

# Write overview to file
path = "".join((folder, "N", str(N), "_", Type, "_L", str(L), "_C",
                str(C), "_T", str(T), "_Overview.txt"))
with open(path, 'w+') as f:
    f.write("> Parameters:\n")

    f.write("N:\t")
    f.write(str(N))
    f.write("\n")

    f.write("Type:\t")
    f.write(Type)
    f.write("\n")

    f.write("Hidden Layers:\t")
    f.write(str(L))
    f.write("\n")

    f.write("Convolution Size:\t")
    f.write(str(C))
    f.write("\n")

    f.write("Trial:\t")
    f.write(str(T))
    f.write("\n")

    f.write("Training Set Size:\t")
    f.write(str(np.shape(X_train)[0]))
    f.write("\n")
```

```
f.write("Validation Set Size:\t")
f.write(str(np.shape(X_test)[0]))
f.write("\n")

f.write("Batch Size:\t")
f.write(str(batch))
f.write("\n")

f.write("Classes:\t")
f.write(str(ncats))
f.write("\n")

f.write("Class Type:\t")
f.write(classtype)
f.write("\n")

f.write("Max Epochs:\t")
f.write(str(max_epoch))
f.write("\n")

f.write("Patience:\t")
f.write(str(patience))
f.write("\n")

f.write("\n")

f.write("> Results:\n")

f.write("Epochs:\t")
f.write(str(np.shape(logTrain.loss)[0]))
f.write("\n")

f.write("Training Time:\t")
f.write(str(training_time))
f.write("\n")

temp = model.evaluate(X_train,y_trainOHE, batch_size = batch)
f.write("Training Evaluation:\n")
```

```
        for ind in range(np.shape(temp)[0]):
            f.write("\t")
            f.write(labels[ind])
            f.write("\t")
            f.write(str(temp[ind]))
            f.write("\n")

        temp = model.evaluate(X_test,y_testOHE, batch_size = batch)
        f.write("Validation Evaluation:\n")
        for ind in range(np.shape(temp)[0]):
            f.write("\t")
            f.write(labels[ind])
            f.write("\t")
            f.write(str(temp[ind]))
            f.write("\n")

        f.write("\n")

        # Model summary
        f.write("> Model Summary")
        f.write("\n")
        with redirect_stdout(f):
            model.summary()

    # Save training metrics
    path = "".join((folder, "N", str(N), "_", Type, "_L", str(L), "_C",
                    str(C), "_T", str(T), "_Training_Metrics.txt"))
    with open(path, 'w+') as f:
        f.write("> Logged Training Metrics\n")

        for ind in range(np.shape(labels)[0]):
            f.write(labels[ind])
            f.write("\t")
        f.write("\n")

        temp = np.stack((logTrain.loss, logTrain.cat_m0,
                         logTrain.cat_m1, logTrain.cat_m2,
                         logTrain.cat_m3, logTrain.cat_m4,
                         logTrain.cat_m5, logTrain.cat_m6,
```

```
                    logTrain.cat_m7, logTrain.cat_m8,
                    logTrain.cat_m9, logTrain.cat_m10,
                    logTrain.cat_top5), axis=1)
    for row in range(np.shape(temp)[0]):
        for col in range(np.shape(temp)[1]):
            f.write(str(temp[row, col]))
            f.write("\t")
        f.write("\n")
    f.write("\n")

# Save validation metrics
path = "".join((folder, "N", str(N), "_", Type, "_L", str(L), "_C",
                str(C), "_T", str(T), "_Validation_Metrics.txt"))
with open(path, 'w+') as f:
    f.write("> Logged Validation Metrics\n")

    for ind in range(np.shape(labels)[0]):
        f.write(labels[ind])
        f.write("\t")
    f.write("\n")

    temp = np.stack((logTest.loss, logTest.cat_m0,
                    logTest.cat_m1, logTest.cat_m2,
                    logTest.cat_m3, logTest.cat_m4,
                    logTest.cat_m5, logTest.cat_m6,
                    logTest.cat_m7, logTest.cat_m8,
                    logTest.cat_m9, logTest.cat_m10,
                    logTest.cat_top5), axis=1)
    for row in range(np.shape(temp)[0]):
        for col in range(np.shape(temp)[1]):
            f.write(str(temp[row, col]))
            f.write("\t")
        f.write("\n")
    f.write("\n")

# Save model
path = "".join((folder, "N", str(N), "_", Type, "_L", str(L), "_C",
                str(C), "_T", str(T), "_model.h5"))
model.save(path)
```

```
    return

def trainNetwork(N, nlayers, csize, ncats, batch, max_epoch, patience,
                 X_train, y_trainOHE, X_test, y_testOHE, trial, hard,
                 ctype, interp, hardness):
    # Make netowrk
    classifier = Sequential()
    classifier.add(layers.Conv2D(csize, (1, 1), activation='relu',
                                 input_shape=(N, N, 2)))
    # Add additional layers
    for cnt in range(nlayers):
        classifier.add(layers.Conv2D(csize, (1, 1), activation='relu'))

    # Add final layers
    classifier.add(layers.Flatten())
    classifier.add(layers.Dense(128, activation='relu'))
    classifier.add(layers.Dense(ncats, activation='softmax'))

    # Compile model
    classifier.compile(optimizer = 'adam',
                       loss = 'categorical_crossentropy',
                       metrics = ['categorical_accuracy', cat_acc_m1,
                                  cat_acc_m2, cat_acc_m3, cat_acc_m4,
                                  cat_acc_m5, cat_acc_m6, cat_acc_m7,
                                  cat_acc_m8, cat_acc_m9, cat_acc_m10,
                                  'top_k_categorical_accuracy'])

    # Log statistics
    logTrain = logTraining()
    logTest = logTesting()

    # Prepare callbacks
    callback = [logTrain, logTest,
                keras.callbacks.EarlyStopping(monitor='val_loss',
                                              min_delta=0,
                                              patience=patience,
                                              verbose=0,
                                              mode='auto')]
```

```
    # Record time and train network
    start_time = time.time()
    classifier.fit(X_train, y_trainOHE, batch_size = batch,
                   epochs = max_epoch, callbacks = callback,
                   validation_data=(X_test, y_testOHE))
    training_time = (time.time() - start_time)

    # Save results
    saveResults(N, nlayers, csize, ncats, batch, max_epoch, patience,
                X_train, y_trainOHE, X_test, y_testOHE, trial, hard,
                ctype, classifier, logTrain, logTest, training_time,
                interp, hardness)
    return


# Main loop
# Train for each <N>
for c_N in range(N_start, N_stop + 1):
    # Train distinguishable
    if easy:
        #Train <trial> times
        for c_T in range(trial_start, trial_stop + 1):
            # Load data
            X, y = loadData(c_N, nsamp, False)
            # Make each sample into a 3D tensor
            X = X.reshape(nsamp, c_N, c_N, 2)

            # Even: sample distribution based classes (don't use)
            if even_distribution:
                classtype = 1
                y = distributionBasedClass(y, nclasses)
            # Default: 0.1 step size classes
            else:
                classtype = 3
                nclasses, y = makeClasses(y, outlierpercentile)
                # Save number of classes
                save_nclasses_easy[c_N - N_start,
                                   c_T - trial_start] = nclasses
```

```
        # Split into training and validation sets
        X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                test_size = vdata)

        # One Hot Encode output
        y_trainOHE = np_utils.to_categorical(y_train)
        y_testOHE = np_utils.to_categorical(y_test)

        # Modify OHE so adjacent values are used by loss function
        if fuzzEncode:
            fuzzOHE(y_trainOHE, spread, factor)
            fuzzOHE(y_testOHE, spread, factor)

        # Train for all combinations of layers and filters
        for c_l in range(l_start, l_stop + 1):
            for c_c in range(0, c_range):
                trainNetwork(c_N, c_l, c_start*(2**c_c), nclasses,
                             batch, max_epoch, patience, X_train,
                             y_trainOHE, X_test, y_testOHE, c_T,
                             False, classtype, False, 0)


# Train indistinguishable
if hard:
    #Train <trial> times
    for c_T in range(trial_start, trial_stop + 1):
        # Load data
        X, y = loadData(c_N, nsamp, True)
        # Make each sample into a 3D tensor
        X = X.reshape(nsamp, c_N, c_N, 2)

        # Even: sample distribution based classes (don't use)
        if even_distribution:
            classtype = 1
            y = distributionBasedClass(y, nclasses)
        #Copy: use same number of classes as distinguishable
        elif easy and copy_easy_cats:
            classtype = 2
            nclasses = save_nclasses_easy[c_N - N_start,
```

```
                                                c_T - trial_start]
                y = cloneClasses(y, outlierpercentile, nclasses)
            # Default: 0.1 step size classes
            else:
                classtype = 3
                nclasses, y = makeClasses(y, outlierpercentile)

            # Split into training and validation sets
            X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = vdata)

            # One Hot Encode output
            y_trainOHE = np_utils.to_categorical(y_train)
            y_testOHE = np_utils.to_categorical(y_test)

            # Modify OHE so adjacent values are used by loss function
            if fuzzEncode:
                fuzzOHE(y_trainOHE, spread, factor)
                fuzzOHE(y_testOHE, spread, factor)

            # Train for all combinations of layers and filters
            for c_l in range(l_start, l_stop + 1):
                for c_c in range(0, c_range):
                    trainNetwork(c_N, c_l, c_start*(2**c_c), nclasses,
                                 batch, max_epoch, patience, X_train,
                                 y_trainOHE, X_test, y_testOHE, c_T,
                                 True, classtype, False, 1)

    # Train partially distinguishable
    if interp:
        for c_T in range(trial_start, trial_stop + 1):
            for c_H in np.nditer(interp_steps, op_flags=['readonly']):
                # Load data
                X, y = loadInterp(c_N, nsamp, c_H)
                # Make each sample into a 3D tensor
                X = X.reshape(nsamp, c_N, c_N, 2)

                # Use 0.1 step size classes
                classtype = 3
```

```
nclasses, y = makeClasses(y, outlierpercentile)

# Split into training and validation sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                        test_size = vdata)

# One Hot Encode output
y_trainOHE = np_utils.to_categorical(y_train)
y_testOHE = np_utils.to_categorical(y_test)

# Modify OHE so adjacent values are used by loss function
if fuzzEncode:
    fuzzOHE(y_trainOHE, spread, factor)
    fuzzOHE(y_testOHE, spread, factor)

# Train for all combinations of layers and filters
for c_l in range(l_start, l_stop + 1):
    for c_c in range(0, c_range):
        trainNetwork(c_N, c_l, c_start*(2**c_c),
                     nclasses, batch, max_epoch,
                     patience, X_train, y_trainOHE,
                     X_test, y_testOHE, c_T, False,
                     classtype, interp, c_H)
```

# Appendix B

# Network Results for Distinguishable Permanents

*Table B.1: Model Metrics for N = 2 (Distinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 59 | 69 | 25.0 | 8,775 |
| 1 | 4 | 59 | 416 | 142.5 | 9,819 |
| 1 | 8 | 59 | 201 | 69.7 | 11,931 |
| 1 | 16 | 59 | 192 | 142.3 | 16,251 |
| 1 | 32 | 59 | 239 | 205.3 | 25,275 |
| 1 | 64 | 59 | 125 | 157.6 | 44,859 |
| 1 | 128 | 59 | 149 | 222.0 | 90,171 |
| 2 | 2 | 59 | 49 | 19.3 | 8,781 |
| 2 | 4 | 59 | 300 | 106.0 | 9,839 |
| 2 | 8 | 59 | 231 | 76.1 | 12,003 |
| 2 | 16 | 59 | 135 | 117.6 | 16,523 |
| 2 | 32 | 59 | 146 | 145.1 | 26,331 |
| 2 | 64 | 59 | 120 | 199.2 | 49,019 |
| 2 | 128 | 59 | 70 | 143.2 | 106,683 |
| 3 | 2 | 59 | 27 | 11.0 | 8,787 |
| 3 | 4 | 59 | 255 | 89.9 | 9,859 |
| 3 | 8 | 59 | 122 | 43.7 | 12,075 |
| 3 | 16 | 59 | 106 | 103.6 | 16,795 |
| 3 | 32 | 59 | 111 | 130.2 | 27,387 |
| 3 | 64 | 59 | 107 | 205.9 | 53,179 |
| 3 | 128 | 59 | 102 | 284.8 | 123,195 |
| 4 | 2 | 59 | 47 | 19.8 | 8,793 |

*Continued on next page*

*Table B.1 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 4 | 4 | 59 | 156 | 61.1 | 9,879 |
| 4 | 8 | 59 | 163 | 62.4 | 12,147 |
| 4 | 16 | 59 | 86 | 86.2 | 17,067 |
| 4 | 32 | 59 | 100 | 121.6 | 28,443 |
| 4 | 64 | 59 | 69 | 172.3 | 57,339 |
| 4 | 128 | 59 | 96 | 331.2 | 139,707 |
| 5 | 2 | 59 | 49 | 22.3 | 8,799 |
| 5 | 4 | 59 | 195 | 80.8 | 9,899 |
| 5 | 8 | 59 | 92 | 38.8 | 12,219 |
| 5 | 16 | 59 | 117 | 162.2 | 17,339 |
| 5 | 32 | 59 | 89 | 150.6 | 29,499 |
| 5 | 64 | 59 | 106 | 329.8 | 61,499 |
| 5 | 128 | 59 | 63 | 283.6 | 156,219 |
| 6 | 2 | 59 | 48 | 23.3 | 8,805 |
| 6 | 4 | 59 | 14 | 8.3 | 9,919 |
| 6 | 8 | 59 | 100 | 42.5 | 12,291 |
| 6 | 16 | 59 | 73 | 111.2 | 17,611 |
| 6 | 32 | 59 | 65 | 113.8 | 30,555 |
| 6 | 64 | 59 | 57 | 194.0 | 65,659 |
| 6 | 128 | 59 | 59 | 309.6 | 172,731 |

*Table B.2: Training Metrics for N = 2 (Distinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.300 | 0.048 | 0.259 | 0.653 | 0.212 |
| 1 | 4 | 3.650 | 0.401 | 0.981 | 1.000 | 0.969 |
| 1 | 8 | 3.282 | 0.536 | 0.995 | 1.000 | 0.992 |
| 1 | 16 | 3.124 | 0.640 | 0.998 | 1.000 | 0.998 |
| 1 | 32 | 3.254 | 0.516 | 0.995 | 1.000 | 0.995 |
| 1 | 64 | 3.279 | 0.520 | 0.995 | 1.000 | 0.988 |
| 1 | 128 | 3.360 | 0.487 | 0.992 | 1.000 | 0.990 |
| 2 | 2 | 7.495 | 0.035 | 0.240 | 0.633 | 0.173 |
| 2 | 4 | 3.436 | 0.443 | 0.994 | 1.000 | 0.988 |
| 2 | 8 | 3.218 | 0.533 | 0.997 | 1.000 | 0.996 |
| 2 | 16 | 3.268 | 0.490 | 0.996 | 1.000 | 0.994 |

*Continued on next page*

*Table B.2 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 2 | 32 | 3.119 | 0.606 | 0.999 | 1.000 | 0.999 |
| 2 | 64 | 3.184 | 0.560 | 0.998 | 1.000 | 0.997 |
| 2 | 128 | 3.197 | 0.519 | 0.999 | 1.000 | 0.995 |
| 3 | 2 | 7.280 | 0.049 | 0.269 | 0.677 | 0.207 |
| 3 | 4 | 3.853 | 0.320 | 0.966 | 0.999 | 0.936 |
| 3 | 8 | 3.375 | 0.463 | 0.992 | 1.000 | 0.985 |
| 3 | 16 | 3.105 | 0.584 | 1.000 | 1.000 | 1.000 |
| 3 | 32 | 3.136 | 0.561 | 0.999 | 1.000 | 0.999 |
| 3 | 64 | 3.184 | 0.534 | 1.000 | 1.000 | 0.999 |
| 3 | 128 | 3.078 | 0.603 | 1.000 | 1.000 | 1.000 |
| 4 | 2 | 7.495 | 0.035 | 0.240 | 0.633 | 0.173 |
| 4 | 4 | 3.755 | 0.359 | 0.973 | 1.000 | 0.956 |
| 4 | 8 | 3.309 | 0.486 | 0.995 | 1.000 | 0.993 |
| 4 | 16 | 3.191 | 0.536 | 0.998 | 1.000 | 0.995 |
| 4 | 32 | 3.086 | 0.662 | 1.000 | 1.000 | 0.998 |
| 4 | 64 | 3.104 | 0.640 | 1.000 | 1.000 | 0.996 |
| 4 | 128 | 3.083 | 0.645 | 1.000 | 1.000 | 1.000 |
| 5 | 2 | 7.495 | 0.035 | 0.240 | 0.633 | 0.173 |
| 5 | 4 | 3.393 | 0.482 | 0.991 | 1.000 | 0.988 |
| 5 | 8 | 3.313 | 0.490 | 0.990 | 1.000 | 0.986 |
| 5 | 16 | 3.155 | 0.550 | 0.999 | 1.000 | 0.998 |
| 5 | 32 | 3.147 | 0.568 | 1.000 | 1.000 | 1.000 |
| 5 | 64 | 3.123 | 0.582 | 1.000 | 1.000 | 0.999 |
| 5 | 128 | 3.158 | 0.559 | 1.000 | 1.000 | 0.998 |
| 6 | 2 | 7.495 | 0.035 | 0.240 | 0.633 | 0.173 |
| 6 | 4 | 7.495 | 0.035 | 0.240 | 0.633 | 0.171 |
| 6 | 8 | 3.140 | 0.574 | 1.000 | 1.000 | 1.000 |
| 6 | 16 | 3.232 | 0.523 | 0.996 | 1.000 | 0.991 |
| 6 | 32 | 3.171 | 0.533 | 1.000 | 1.000 | 0.998 |
| 6 | 64 | 3.280 | 0.462 | 0.998 | 1.000 | 0.997 |
| 6 | 128 | 3.255 | 0.488 | 0.999 | 1.000 | 0.997 |

*Table B.3: Validation Metrics for N = 2 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.370 | 0.035 | 0.247 | 0.645 | 0.180 |
| 1 | 4 | 3.832 | 0.308 | 0.965 | 1.000 | 0.924 |
| 1 | 8 | 3.423 | 0.435 | 0.984 | 1.000 | 0.967 |
| 1 | 16 | 3.210 | 0.533 | 0.996 | 1.000 | 0.989 |
| 1 | 32 | 3.403 | 0.435 | 0.994 | 1.000 | 0.973 |
| 1 | 64 | 3.405 | 0.408 | 0.988 | 1.000 | 0.972 |
| 1 | 128 | 3.498 | 0.380 | 0.981 | 1.000 | 0.968 |
| 2 | 2 | 7.544 | 0.032 | 0.223 | 0.616 | 0.170 |
| 2 | 4 | 3.559 | 0.364 | 0.982 | 1.000 | 0.965 |
| 2 | 8 | 3.321 | 0.461 | 0.990 | 1.000 | 0.988 |
| 2 | 16 | 3.324 | 0.445 | 0.993 | 1.000 | 0.987 |
| 2 | 32 | 3.194 | 0.514 | 0.997 | 1.000 | 0.995 |
| 2 | 64 | 3.268 | 0.497 | 0.994 | 1.000 | 0.991 |
| 2 | 128 | 3.269 | 0.461 | 0.999 | 1.000 | 0.988 |
| 3 | 2 | 7.357 | 0.043 | 0.262 | 0.658 | 0.192 |
| 3 | 4 | 3.968 | 0.266 | 0.955 | 1.000 | 0.908 |
| 3 | 8 | 3.478 | 0.396 | 0.988 | 1.000 | 0.967 |
| 3 | 16 | 3.160 | 0.537 | 0.998 | 1.000 | 0.997 |
| 3 | 32 | 3.193 | 0.511 | 0.999 | 1.000 | 0.993 |
| 3 | 64 | 3.233 | 0.489 | 0.999 | 1.000 | 0.995 |
| 3 | 128 | 3.139 | 0.517 | 1.000 | 1.000 | 0.997 |
| 4 | 2 | 7.544 | 0.032 | 0.223 | 0.616 | 0.170 |
| 4 | 4 | 3.872 | 0.279 | 0.964 | 0.999 | 0.931 |
| 4 | 8 | 3.418 | 0.418 | 0.989 | 1.000 | 0.975 |
| 4 | 16 | 3.269 | 0.440 | 0.998 | 1.000 | 0.989 |
| 4 | 32 | 3.136 | 0.603 | 1.000 | 1.000 | 0.995 |
| 4 | 64 | 3.169 | 0.549 | 0.999 | 1.000 | 0.993 |
| 4 | 128 | 3.144 | 0.570 | 1.000 | 1.000 | 0.999 |
| 5 | 2 | 7.544 | 0.032 | 0.223 | 0.616 | 0.170 |
| 5 | 4 | 3.503 | 0.392 | 0.989 | 1.000 | 0.961 |
| 5 | 8 | 3.398 | 0.430 | 0.987 | 1.000 | 0.965 |
| 5 | 16 | 3.236 | 0.471 | 0.997 | 1.000 | 0.992 |
| 5 | 32 | 3.225 | 0.477 | 1.000 | 1.000 | 0.997 |
| 5 | 64 | 3.180 | 0.516 | 0.999 | 1.000 | 0.996 |
| 5 | 128 | 3.231 | 0.472 | 0.998 | 1.000 | 0.993 |

*Continued on next page*

*Table B.3 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 2 | 7.544 | 0.032 | 0.223 | 0.616 | 0.170 |
| 6 | 4 | 7.545 | 0.032 | 0.223 | 0.616 | 0.159 |
| 6 | 8 | 3.216 | 0.493 | 0.999 | 1.000 | 0.995 |
| 6 | 16 | 3.321 | 0.438 | 0.995 | 1.000 | 0.980 |
| 6 | 32 | 3.252 | 0.456 | 0.998 | 1.000 | 0.994 |
| 6 | 64 | 3.370 | 0.399 | 0.994 | 1.000 | 0.989 |
| 6 | 128 | 3.337 | 0.413 | 0.997 | 1.000 | 0.993 |

*Table B.4: Model Metrics for N = 3 (Distinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 56 | 98 | 45.2 | 9,668 |
| 1 | 4 | 56 | 289 | 138.2 | 11,992 |
| 1 | 8 | 56 | 263 | 135.2 | 16,664 |
| 1 | 16 | 56 | 219 | 426.2 | 26,104 |
| 1 | 32 | 56 | 223 | 532.2 | 45,368 |
| 1 | 64 | 56 | 144 | 400.2 | 85,432 |
| 1 | 128 | 56 | 111 | 364.4 | 171,704 |
| 2 | 2 | 56 | 39 | 23.7 | 9,674 |
| 2 | 4 | 56 | 197 | 107.4 | 12,012 |
| 2 | 8 | 56 | 152 | 92.4 | 16,736 |
| 2 | 16 | 56 | 162 | 374.5 | 26,376 |
| 2 | 32 | 56 | 126 | 364.2 | 46,424 |
| 2 | 64 | 56 | 151 | 517.9 | 89,592 |
| 2 | 128 | 56 | 130 | 587.7 | 188,216 |
| 3 | 2 | 56 | 63 | 38.1 | 9,680 |
| 3 | 4 | 56 | 113 | 63.7 | 12,032 |
| 3 | 8 | 56 | 197 | 134.9 | 16,808 |
| 3 | 16 | 56 | 155 | 404.4 | 26,648 |
| 3 | 32 | 56 | 156 | 532.2 | 47,480 |
| 3 | 64 | 56 | 167 | 657.2 | 93,752 |
| 3 | 128 | 56 | 147 | 801.3 | 204,728 |
| 4 | 2 | 56 | 118 | 92.7 | 9,686 |
| 4 | 4 | 56 | 19 | 21.5 | 12,052 |
| 4 | 8 | 56 | 177 | 137.0 | 16,880 |

*Continued on next page*

*Table B.4 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 4 | 16 | 56 | 146 | 416.0 | 26,920 |
| 4 | 32 | 56 | 74 | 291.6 | 48,536 |
| 4 | 64 | 56 | 99 | 516.1 | 97,912 |
| 4 | 128 | 56 | 187 | 1246.5 | 221,240 |
| 5 | 2 | 56 | 77 | 53.8 | 9,692 |
| 5 | 4 | 56 | 265 | 185.2 | 12,072 |
| 5 | 8 | 56 | 169 | 140.7 | 16,952 |
| 5 | 16 | 56 | 111 | 367.5 | 27,192 |
| 5 | 32 | 56 | 160 | 771.8 | 49,592 |
| 5 | 64 | 56 | 152 | 867.8 | 102,072 |
| 5 | 128 | 56 | 77 | 610.1 | 237,752 |
| 6 | 2 | 56 | 80 | 60.6 | 9,698 |
| 6 | 4 | 56 | 92 | 72.1 | 12,092 |
| 6 | 8 | 56 | 222 | 194.1 | 17,024 |
| 6 | 16 | 56 | 167 | 592.3 | 27,464 |
| 6 | 32 | 56 | 165 | 873.4 | 50,648 |
| 6 | 64 | 56 | 209 | 1447.5 | 106,232 |
| 6 | 128 | 56 | 93 | 853.4 | 254,264 |

*Table B.5: Training Metrics for N = 3 (Distinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.423 | 0.037 | 0.244 | 0.654 | 0.178 |
| 1 | 4 | 4.208 | 0.357 | 0.899 | 0.999 | 0.882 |
| 1 | 8 | 3.868 | 0.439 | 0.945 | 1.000 | 0.936 |
| 1 | 16 | 3.924 | 0.407 | 0.940 | 0.999 | 0.926 |
| 1 | 32 | 3.672 | 0.468 | 0.971 | 1.000 | 0.964 |
| 1 | 64 | 4.114 | 0.320 | 0.913 | 0.999 | 0.883 |
| 1 | 128 | 3.935 | 0.370 | 0.940 | 1.000 | 0.915 |
| 2 | 2 | 7.215 | 0.054 | 0.272 | 0.683 | 0.231 |
| 2 | 4 | 4.503 | 0.292 | 0.831 | 0.998 | 0.794 |
| 2 | 8 | 3.887 | 0.382 | 0.947 | 1.000 | 0.925 |
| 2 | 16 | 3.604 | 0.455 | 0.977 | 1.000 | 0.969 |
| 2 | 32 | 3.965 | 0.338 | 0.943 | 1.000 | 0.911 |
| 2 | 64 | 3.786 | 0.372 | 0.965 | 1.000 | 0.942 |

*Continued on next page*

*Table B.5 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 2 | 128 | 3.818 | 0.385 | 0.956 | 1.000 | 0.929 |
| 3 | 2 | 6.717 | 0.090 | 0.358 | 0.770 | 0.327 |
| 3 | 4 | 6.368 | 0.105 | 0.414 | 0.855 | 0.375 |
| 3 | 8 | 3.753 | 0.390 | 0.961 | 1.000 | 0.952 |
| 3 | 16 | 3.719 | 0.392 | 0.968 | 1.000 | 0.954 |
| 3 | 32 | 3.702 | 0.413 | 0.970 | 1.000 | 0.954 |
| 3 | 64 | 3.793 | 0.355 | 0.965 | 1.000 | 0.942 |
| 3 | 128 | 3.417 | 0.486 | 0.994 | 1.000 | 0.987 |
| 4 | 2 | 7.423 | 0.037 | 0.244 | 0.654 | 0.178 |
| 4 | 4 | 7.423 | 0.037 | 0.242 | 0.651 | 0.178 |
| 4 | 8 | 3.691 | 0.390 | 0.972 | 1.000 | 0.958 |
| 4 | 16 | 3.812 | 0.362 | 0.962 | 0.999 | 0.940 |
| 4 | 32 | 4.060 | 0.326 | 0.921 | 0.999 | 0.887 |
| 4 | 64 | 3.463 | 0.444 | 0.990 | 1.000 | 0.987 |
| 4 | 128 | 3.307 | 0.521 | 0.996 | 1.000 | 0.996 |
| 5 | 2 | 7.423 | 0.037 | 0.244 | 0.654 | 0.178 |
| 5 | 4 | 3.811 | 0.408 | 0.956 | 1.000 | 0.944 |
| 5 | 8 | 3.800 | 0.364 | 0.959 | 1.000 | 0.939 |
| 5 | 16 | 3.583 | 0.427 | 0.982 | 1.000 | 0.975 |
| 5 | 32 | 3.560 | 0.422 | 0.985 | 1.000 | 0.981 |
| 5 | 64 | 4.007 | 0.291 | 0.954 | 1.000 | 0.918 |
| 5 | 128 | 3.574 | 0.395 | 0.987 | 1.000 | 0.979 |
| 6 | 2 | 7.423 | 0.037 | 0.244 | 0.654 | 0.178 |
| 6 | 4 | 4.406 | 0.277 | 0.860 | 0.997 | 0.812 |
| 6 | 8 | 3.563 | 0.455 | 0.983 | 1.000 | 0.974 |
| 6 | 16 | 3.604 | 0.432 | 0.983 | 1.000 | 0.974 |
| 6 | 32 | 3.423 | 0.465 | 0.993 | 1.000 | 0.988 |
| 6 | 64 | 3.795 | 0.351 | 0.967 | 1.000 | 0.947 |
| 6 | 128 | 3.879 | 0.356 | 0.947 | 1.000 | 0.919 |

*Table B.6: Validation Metrics for N = 3 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.465 | 0.027 | 0.236 | 0.635 | 0.163 |
| 1 | 4 | 4.703 | 0.173 | 0.821 | 0.997 | 0.707 |

*Continued on next page*

*Table B.6 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 8 | 4.558 | 0.206 | 0.859 | 0.998 | 0.774 |
| 1 | 16 | 4.417 | 0.214 | 0.885 | 0.998 | 0.782 |
| 1 | 32 | 4.262 | 0.248 | 0.915 | 0.998 | 0.838 |
| 1 | 64 | 4.432 | 0.205 | 0.880 | 0.999 | 0.789 |
| 1 | 128 | 4.352 | 0.223 | 0.903 | 0.999 | 0.810 |
| 2 | 2 | 7.301 | 0.031 | 0.253 | 0.648 | 0.186 |
| 2 | 4 | 4.948 | 0.141 | 0.750 | 0.996 | 0.649 |
| 2 | 8 | 4.237 | 0.246 | 0.912 | 1.000 | 0.837 |
| 2 | 16 | 3.966 | 0.271 | 0.941 | 1.000 | 0.902 |
| 2 | 32 | 4.270 | 0.223 | 0.916 | 0.999 | 0.836 |
| 2 | 64 | 4.076 | 0.254 | 0.944 | 1.000 | 0.871 |
| 2 | 128 | 4.351 | 0.222 | 0.910 | 0.997 | 0.817 |
| 3 | 2 | 6.865 | 0.061 | 0.317 | 0.750 | 0.252 |
| 3 | 4 | 6.552 | 0.071 | 0.380 | 0.836 | 0.285 |
| 3 | 8 | 4.120 | 0.264 | 0.935 | 1.000 | 0.874 |
| 3 | 16 | 3.968 | 0.278 | 0.956 | 1.000 | 0.898 |
| 3 | 32 | 4.057 | 0.256 | 0.946 | 1.000 | 0.884 |
| 3 | 64 | 4.098 | 0.262 | 0.938 | 0.999 | 0.866 |
| 3 | 128 | 3.911 | 0.301 | 0.952 | 1.000 | 0.916 |
| 4 | 2 | 7.465 | 0.027 | 0.236 | 0.635 | 0.163 |
| 4 | 4 | 7.466 | 0.032 | 0.229 | 0.630 | 0.163 |
| 4 | 8 | 4.056 | 0.247 | 0.944 | 1.000 | 0.888 |
| 4 | 16 | 4.070 | 0.261 | 0.942 | 0.999 | 0.889 |
| 4 | 32 | 4.318 | 0.240 | 0.894 | 0.999 | 0.813 |
| 4 | 64 | 3.715 | 0.330 | 0.972 | 1.000 | 0.940 |
| 4 | 128 | 3.632 | 0.348 | 0.982 | 1.000 | 0.962 |
| 5 | 2 | 7.465 | 0.027 | 0.236 | 0.635 | 0.163 |
| 5 | 4 | 4.415 | 0.221 | 0.895 | 0.999 | 0.815 |
| 5 | 8 | 4.128 | 0.238 | 0.922 | 1.000 | 0.856 |
| 5 | 16 | 3.890 | 0.296 | 0.962 | 1.000 | 0.923 |
| 5 | 32 | 3.879 | 0.294 | 0.960 | 1.000 | 0.921 |
| 5 | 64 | 4.298 | 0.226 | 0.928 | 1.000 | 0.855 |
| 5 | 128 | 3.837 | 0.284 | 0.968 | 1.000 | 0.928 |
| 6 | 2 | 7.465 | 0.027 | 0.236 | 0.635 | 0.163 |
| 6 | 4 | 4.742 | 0.155 | 0.825 | 0.999 | 0.706 |
| 6 | 8 | 4.015 | 0.287 | 0.942 | 1.000 | 0.900 |

*Continued on next page*

*Table B.6 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 16 | 4.026 | 0.263 | 0.953 | 0.999 | 0.895 |
| 6 | 32 | 3.687 | 0.336 | 0.974 | 0.999 | 0.939 |
| 6 | 64 | 4.233 | 0.226 | 0.925 | 0.999 | 0.860 |
| 6 | 128 | 4.324 | 0.221 | 0.890 | 0.999 | 0.808 |

*Table B.7: Model Metrics for N = 4 (Distinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 56 | 33 | 34.1 | 11,460 |
| 1 | 4 | 56 | 49 | 62.7 | 15,576 |
| 1 | 8 | 56 | 45 | 55.5 | 23,832 |
| 1 | 16 | 56 | 47 | 18.7 | 40,440 |
| 1 | 32 | 56 | 30 | 17.8 | 74,040 |
| 1 | 64 | 56 | 23 | 19.1 | 142,776 |
| 1 | 128 | 56 | 21 | 33.7 | 286,392 |
| 2 | 2 | 56 | 61 | 63.4 | 11,466 |
| 2 | 4 | 56 | 78 | 94.1 | 15,596 |
| 2 | 8 | 56 | 175 | 233.3 | 23,904 |
| 2 | 16 | 56 | 35 | 18.8 | 40,712 |
| 2 | 32 | 56 | 26 | 18.1 | 75,096 |
| 2 | 64 | 56 | 123 | 139.5 | 146,936 |
| 2 | 128 | 56 | 83 | 156.1 | 302,904 |
| 3 | 2 | 56 | 23 | 33.0 | 11,472 |
| 3 | 4 | 56 | 74 | 95.1 | 15,616 |
| 3 | 8 | 56 | 155 | 241.3 | 23,976 |
| 3 | 16 | 56 | 140 | 74.2 | 40,984 |
| 3 | 32 | 56 | 137 | 100.3 | 76,152 |
| 3 | 64 | 56 | 95 | 117.1 | 151,096 |
| 3 | 128 | 56 | 65 | 162.2 | 319,416 |
| 4 | 2 | 56 | 68 | 79.6 | 11,478 |
| 4 | 4 | 56 | 59 | 93.2 | 15,636 |
| 4 | 8 | 56 | 147 | 255.4 | 24,048 |
| 4 | 16 | 56 | 98 | 62.5 | 41,256 |
| 4 | 32 | 56 | 117 | 96.9 | 77,208 |
| 4 | 64 | 56 | 97 | 129.6 | 155,256 |

*Continued on next page*

*Table B.7 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 4 | 128 | 56 | 92 | 287.7 | 335,928 |
| 5 | 2 | 56 | 60 | 78.6 | 11,484 |
| 5 | 4 | 56 | 246 | 402.0 | 15,656 |
| 5 | 8 | 56 | 127 | 252.3 | 24,120 |
| 5 | 16 | 56 | 161 | 425.0 | 41,528 |
| 5 | 32 | 56 | 107 | 342.1 | 78,264 |
| 5 | 64 | 56 | 83 | 311.1 | 159,416 |
| 5 | 128 | 56 | 86 | 231.7 | 352,440 |
| 6 | 2 | 56 | 61 | 85.1 | 11,490 |
| 6 | 4 | 56 | 47 | 100.7 | 15,676 |
| 6 | 8 | 56 | 143 | 320.3 | 24,192 |
| 6 | 16 | 56 | 124 | 411.8 | 41,800 |
| 6 | 32 | 56 | 110 | 418.0 | 79,320 |
| 6 | 64 | 56 | 105 | 459.9 | 163,576 |
| 6 | 128 | 56 | 77 | 234.1 | 368,952 |

*Table B.8: Training Metrics for N = 4 (Distinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.202 | 0.054 | 0.264 | 0.669 | 0.231 |
| 1 | 4 | 5.666 | 0.147 | 0.563 | 0.956 | 0.518 |
| 1 | 8 | 5.519 | 0.178 | 0.610 | 0.966 | 0.568 |
| 1 | 16 | 5.488 | 0.176 | 0.617 | 0.968 | 0.570 |
| 1 | 32 | 5.515 | 0.169 | 0.606 | 0.968 | 0.553 |
| 1 | 64 | 5.564 | 0.150 | 0.592 | 0.967 | 0.538 |
| 1 | 128 | 5.638 | 0.151 | 0.567 | 0.957 | 0.518 |
| 2 | 2 | 6.761 | 0.075 | 0.328 | 0.763 | 0.293 |
| 2 | 4 | 6.593 | 0.101 | 0.365 | 0.780 | 0.357 |
| 2 | 8 | 4.194 | 0.375 | 0.887 | 0.999 | 0.874 |
| 2 | 16 | 5.558 | 0.151 | 0.611 | 0.970 | 0.536 |
| 2 | 32 | 5.579 | 0.144 | 0.593 | 0.970 | 0.519 |
| 2 | 64 | 4.060 | 0.378 | 0.916 | 1.000 | 0.891 |
| 2 | 128 | 4.692 | 0.251 | 0.770 | 0.994 | 0.732 |
| 3 | 2 | 7.420 | 0.037 | 0.236 | 0.631 | 0.174 |
| 3 | 4 | 5.690 | 0.150 | 0.554 | 0.958 | 0.498 |

*Continued on next page*

*Table B.8 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 3 | 8 | 4.258 | 0.330 | 0.888 | 0.999 | 0.852 |
| 3 | 16 | 4.087 | 0.374 | 0.904 | 1.000 | 0.889 |
| 3 | 32 | 4.223 | 0.299 | 0.892 | 0.999 | 0.854 |
| 3 | 64 | 4.296 | 0.291 | 0.879 | 0.999 | 0.837 |
| 3 | 128 | 4.404 | 0.292 | 0.854 | 0.998 | 0.805 |
| 4 | 2 | 7.420 | 0.037 | 0.236 | 0.631 | 0.177 |
| 4 | 4 | 6.739 | 0.116 | 0.368 | 0.722 | 0.378 |
| 4 | 8 | 4.269 | 0.311 | 0.891 | 0.999 | 0.850 |
| 4 | 16 | 4.338 | 0.300 | 0.867 | 0.999 | 0.827 |
| 4 | 32 | 4.145 | 0.336 | 0.904 | 1.000 | 0.874 |
| 4 | 64 | 4.340 | 0.289 | 0.867 | 0.999 | 0.822 |
| 4 | 128 | 4.153 | 0.324 | 0.908 | 0.999 | 0.862 |
| 5 | 2 | 7.420 | 0.037 | 0.236 | 0.631 | 0.174 |
| 5 | 4 | 4.224 | 0.372 | 0.891 | 0.999 | 0.868 |
| 5 | 8 | 4.332 | 0.310 | 0.872 | 0.999 | 0.834 |
| 5 | 16 | 4.221 | 0.299 | 0.897 | 0.999 | 0.855 |
| 5 | 32 | 4.115 | 0.348 | 0.907 | 1.000 | 0.879 |
| 5 | 64 | 4.395 | 0.287 | 0.857 | 0.999 | 0.809 |
| 5 | 128 | 4.178 | 0.321 | 0.894 | 1.000 | 0.860 |
| 6 | 2 | 7.420 | 0.037 | 0.236 | 0.631 | 0.177 |
| 6 | 4 | 6.714 | 0.087 | 0.340 | 0.759 | 0.315 |
| 6 | 8 | 4.153 | 0.333 | 0.906 | 0.999 | 0.868 |
| 6 | 16 | 4.300 | 0.278 | 0.886 | 0.999 | 0.836 |
| 6 | 32 | 4.311 | 0.283 | 0.877 | 1.000 | 0.827 |
| 6 | 64 | 4.326 | 0.266 | 0.876 | 1.000 | 0.824 |
| 6 | 128 | 4.357 | 0.268 | 0.870 | 0.999 | 0.814 |

*Table B.9: Validation Metrics for N = 4 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.282 | 0.037 | 0.259 | 0.665 | 0.201 |
| 1 | 4 | 5.805 | 0.092 | 0.539 | 0.960 | 0.413 |
| 1 | 8 | 5.725 | 0.093 | 0.573 | 0.960 | 0.433 |
| 1 | 16 | 5.741 | 0.101 | 0.575 | 0.966 | 0.447 |
| 1 | 32 | 5.751 | 0.103 | 0.563 | 0.964 | 0.423 |

*Continued on next page*

*Table B.9 – continued from previous page*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 64 | 5.764 | 0.099 | 0.557 | 0.957 | 0.429 |
| 1 | 128 | 5.862 | 0.087 | 0.522 | 0.953 | 0.394 |
| 2 | 2 | 6.811 | 0.057 | 0.330 | 0.773 | 0.253 |
| 2 | 4 | 6.815 | 0.054 | 0.332 | 0.770 | 0.241 |
| 2 | 8 | 5.059 | 0.143 | 0.750 | 0.994 | 0.627 |
| 2 | 16 | 5.728 | 0.112 | 0.581 | 0.967 | 0.452 |
| 2 | 32 | 5.771 | 0.092 | 0.568 | 0.961 | 0.425 |
| 2 | 64 | 5.058 | 0.152 | 0.789 | 0.991 | 0.668 |
| 2 | 128 | 5.509 | 0.116 | 0.671 | 0.988 | 0.534 |
| 3 | 2 | 7.429 | 0.036 | 0.237 | 0.638 | 0.173 |
| 3 | 4 | 5.875 | 0.086 | 0.514 | 0.955 | 0.379 |
| 3 | 8 | 4.722 | 0.175 | 0.823 | 0.998 | 0.703 |
| 3 | 16 | 4.809 | 0.146 | 0.820 | 0.997 | 0.693 |
| 3 | 32 | 4.658 | 0.180 | 0.828 | 0.998 | 0.730 |
| 3 | 64 | 4.801 | 0.161 | 0.802 | 0.994 | 0.678 |
| 3 | 128 | 4.906 | 0.164 | 0.781 | 0.993 | 0.662 |
| 4 | 2 | 7.428 | 0.036 | 0.237 | 0.638 | 0.175 |
| 4 | 4 | 7.137 | 0.042 | 0.283 | 0.699 | 0.211 |
| 4 | 8 | 4.681 | 0.172 | 0.827 | 0.999 | 0.715 |
| 4 | 16 | 4.898 | 0.153 | 0.803 | 0.995 | 0.669 |
| 4 | 32 | 4.740 | 0.174 | 0.828 | 0.998 | 0.715 |
| 4 | 64 | 4.936 | 0.153 | 0.801 | 0.996 | 0.662 |
| 4 | 128 | 4.871 | 0.158 | 0.794 | 0.997 | 0.666 |
| 5 | 2 | 7.428 | 0.036 | 0.237 | 0.638 | 0.173 |
| 5 | 4 | 4.809 | 0.147 | 0.799 | 0.997 | 0.679 |
| 5 | 8 | 4.848 | 0.148 | 0.807 | 0.998 | 0.681 |
| 5 | 16 | 4.583 | 0.196 | 0.860 | 0.997 | 0.747 |
| 5 | 32 | 4.737 | 0.186 | 0.831 | 0.995 | 0.721 |
| 5 | 64 | 4.864 | 0.156 | 0.801 | 0.997 | 0.681 |
| 5 | 128 | 5.111 | 0.153 | 0.768 | 0.994 | 0.644 |
| 6 | 2 | 7.428 | 0.036 | 0.237 | 0.638 | 0.175 |
| 6 | 4 | 6.856 | 0.050 | 0.314 | 0.760 | 0.231 |
| 6 | 8 | 4.689 | 0.182 | 0.835 | 0.997 | 0.708 |
| 6 | 16 | 4.700 | 0.160 | 0.849 | 0.998 | 0.734 |
| 6 | 32 | 4.758 | 0.178 | 0.820 | 0.996 | 0.705 |
| 6 | 64 | 4.737 | 0.172 | 0.821 | 0.995 | 0.693 |

*Continued on next page*

*Table B.9 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 128 | 4.913 | 0.153 | 0.783 | 0.995 | 0.669 |

*Table B.10: Model Metrics for N = 5 (Distinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 52 | 57 | 107.8 | 13,248 |
| 1 | 4 | 52 | 68 | 173.8 | 19,668 |
| 1 | 8 | 52 | 43 | 137.8 | 32,532 |
| 1 | 16 | 52 | 35 | 26.7 | 58,356 |
| 1 | 32 | 52 | 29 | 27.1 | 110,388 |
| 1 | 64 | 52 | 26 | 33.9 | 215,988 |
| 1 | 128 | 52 | 25 | 68.6 | 433,332 |
| 2 | 2 | 52 | 60 | 118.8 | 13,254 |
| 2 | 4 | 52 | 75 | 182.2 | 19,688 |
| 2 | 8 | 52 | 41 | 141.8 | 32,604 |
| 2 | 16 | 52 | 26 | 21.5 | 58,628 |
| 2 | 32 | 52 | 28 | 27.4 | 111,444 |
| 2 | 64 | 52 | 129 | 183.1 | 220,148 |
| 2 | 128 | 52 | 29 | 103.1 | 449,844 |
| 3 | 2 | 52 | 96 | 186.6 | 13,260 |
| 3 | 4 | 52 | 247 | 633.7 | 19,708 |
| 3 | 8 | 52 | 49 | 187.8 | 32,676 |
| 3 | 16 | 52 | 157 | 125.7 | 58,900 |
| 3 | 32 | 52 | 109 | 134.9 | 112,500 |
| 3 | 64 | 52 | 30 | 67.6 | 224,308 |
| 3 | 128 | 52 | 26 | 112.1 | 466,356 |
| 4 | 2 | 52 | 81 | 165.6 | 13,266 |
| 4 | 4 | 52 | 13 | 50.0 | 19,728 |
| 4 | 8 | 52 | 168 | 622.5 | 32,748 |
| 4 | 16 | 52 | 114 | 119.7 | 59,172 |
| 4 | 32 | 52 | 74 | 104.7 | 113,556 |
| 4 | 64 | 52 | 46 | 98.9 | 228,468 |
| 4 | 128 | 52 | 57 | 269.0 | 482,868 |
| 5 | 2 | 52 | 78 | 181.8 | 13,272 |
| 5 | 4 | 52 | 32 | 112.8 | 19,748 |

*Continued on next page*

*Table B.10 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 5 | 8 | 52 | 162 | 763.8 | 32,820 |
| 5 | 16 | 52 | 127 | 485.2 | 59,444 |
| 5 | 32 | 52 | 133 | 643.9 | 114,612 |
| 5 | 64 | 52 | 38 | 199.5 | 232,628 |
| 5 | 128 | 52 | 88 | 410.0 | 499,380 |
| 6 | 2 | 52 | 79 | 198.9 | 13,278 |
| 6 | 4 | 52 | 161 | 565.5 | 19,768 |
| 6 | 8 | 52 | 164 | 856.1 | 32,892 |
| 6 | 16 | 52 | 135 | 572.8 | 59,716 |
| 6 | 32 | 52 | 111 | 625.1 | 115,668 |
| 6 | 64 | 52 | 86 | 513.5 | 236,788 |
| 6 | 128 | 52 | 96 | 472.0 | 515,892 |

*Table B.11: Training Metrics for N = 5 (Distinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 6.566 | 0.112 | 0.368 | 0.785 | 0.379 |
| 1 | 4 | 5.348 | 0.246 | 0.635 | 0.964 | 0.644 |
| 1 | 8 | 5.414 | 0.188 | 0.627 | 0.973 | 0.586 |
| 1 | 16 | 5.347 | 0.213 | 0.638 | 0.975 | 0.610 |
| 1 | 32 | 5.481 | 0.196 | 0.610 | 0.969 | 0.565 |
| 1 | 64 | 5.363 | 0.199 | 0.634 | 0.974 | 0.603 |
| 1 | 128 | 5.499 | 0.191 | 0.582 | 0.960 | 0.569 |
| 2 | 2 | 6.570 | 0.103 | 0.374 | 0.790 | 0.367 |
| 2 | 4 | 5.483 | 0.153 | 0.615 | 0.973 | 0.538 |
| 2 | 8 | 5.389 | 0.179 | 0.638 | 0.976 | 0.576 |
| 2 | 16 | 5.440 | 0.171 | 0.631 | 0.975 | 0.563 |
| 2 | 32 | 5.373 | 0.196 | 0.635 | 0.975 | 0.594 |
| 2 | 64 | 4.096 | 0.395 | 0.905 | 0.999 | 0.883 |
| 2 | 128 | 5.326 | 0.196 | 0.644 | 0.977 | 0.608 |
| 3 | 2 | 7.322 | 0.039 | 0.254 | 0.666 | 0.187 |
| 3 | 4 | 4.166 | 0.389 | 0.896 | 0.999 | 0.876 |
| 3 | 8 | 5.389 | 0.180 | 0.648 | 0.976 | 0.587 |
| 3 | 16 | 4.200 | 0.352 | 0.888 | 0.999 | 0.855 |
| 3 | 32 | 4.388 | 0.308 | 0.852 | 0.998 | 0.815 |

*Continued on next page*

*Table B.11 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 3 | 64 | 5.368 | 0.185 | 0.637 | 0.975 | 0.582 |
| 3 | 128 | 5.277 | 0.222 | 0.654 | 0.976 | 0.631 |
| 4 | 2 | 7.322 | 0.039 | 0.254 | 0.666 | 0.187 |
| 4 | 4 | 7.323 | 0.038 | 0.252 | 0.662 | 0.180 |
| 4 | 8 | 4.206 | 0.337 | 0.892 | 1.000 | 0.859 |
| 4 | 16 | 4.211 | 0.387 | 0.886 | 0.998 | 0.875 |
| 4 | 32 | 4.797 | 0.262 | 0.753 | 0.991 | 0.726 |
| 4 | 64 | 5.227 | 0.216 | 0.637 | 0.979 | 0.616 |
| 4 | 128 | 4.406 | 0.331 | 0.844 | 0.997 | 0.829 |
| 5 | 2 | 7.322 | 0.039 | 0.254 | 0.666 | 0.187 |
| 5 | 4 | 7.059 | 0.071 | 0.300 | 0.680 | 0.271 |
| 5 | 8 | 4.313 | 0.305 | 0.876 | 0.999 | 0.834 |
| 5 | 16 | 4.165 | 0.339 | 0.900 | 1.000 | 0.863 |
| 5 | 32 | 4.288 | 0.299 | 0.878 | 0.999 | 0.831 |
| 5 | 64 | 4.963 | 0.261 | 0.722 | 0.983 | 0.705 |
| 5 | 128 | 4.277 | 0.311 | 0.877 | 0.999 | 0.840 |
| 6 | 2 | 7.322 | 0.039 | 0.254 | 0.666 | 0.187 |
| 6 | 4 | 4.418 | 0.352 | 0.849 | 0.998 | 0.840 |
| 6 | 8 | 4.160 | 0.344 | 0.895 | 1.000 | 0.860 |
| 6 | 16 | 4.159 | 0.333 | 0.907 | 1.000 | 0.865 |
| 6 | 32 | 4.595 | 0.237 | 0.817 | 0.998 | 0.745 |
| 6 | 64 | 4.464 | 0.295 | 0.829 | 0.998 | 0.800 |
| 6 | 128 | 4.114 | 0.346 | 0.911 | 1.000 | 0.871 |

*Table B.12: Validation Metrics for N = 5 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 6.859 | 0.049 | 0.289 | 0.771 | 0.228 |
| 1 | 4 | 5.862 | 0.091 | 0.551 | 0.965 | 0.406 |
| 1 | 8 | 5.617 | 0.101 | 0.595 | 0.978 | 0.446 |
| 1 | 16 | 5.622 | 0.111 | 0.574 | 0.974 | 0.462 |
| 1 | 32 | 5.774 | 0.095 | 0.554 | 0.969 | 0.424 |
| 1 | 64 | 5.702 | 0.104 | 0.582 | 0.976 | 0.447 |
| 1 | 128 | 5.841 | 0.087 | 0.516 | 0.959 | 0.400 |

*Continued on next page*

*Table B.12 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 2 | 2 | 6.814 | 0.053 | 0.321 | 0.779 | 0.238 |
| 2 | 4 | 5.560 | 0.101 | 0.614 | 0.978 | 0.470 |
| 2 | 8 | 5.563 | 0.108 | 0.603 | 0.976 | 0.477 |
| 2 | 16 | 5.612 | 0.113 | 0.591 | 0.972 | 0.471 |
| 2 | 32 | 5.649 | 0.101 | 0.594 | 0.975 | 0.450 |
| 2 | 64 | 5.244 | 0.142 | 0.736 | 0.992 | 0.608 |
| 2 | 128 | 5.663 | 0.106 | 0.584 | 0.969 | 0.448 |
| 3 | 2 | 7.327 | 0.031 | 0.235 | 0.662 | 0.166 |
| 3 | 4 | 5.054 | 0.146 | 0.766 | 0.994 | 0.634 |
| 3 | 8 | 5.554 | 0.111 | 0.613 | 0.980 | 0.463 |
| 3 | 16 | 4.990 | 0.160 | 0.766 | 0.998 | 0.664 |
| 3 | 32 | 5.214 | 0.143 | 0.738 | 0.996 | 0.601 |
| 3 | 64 | 5.699 | 0.100 | 0.569 | 0.968 | 0.451 |
| 3 | 128 | 5.664 | 0.102 | 0.591 | 0.970 | 0.432 |
| 4 | 2 | 7.327 | 0.031 | 0.235 | 0.662 | 0.166 |
| 4 | 4 | 7.328 | 0.034 | 0.241 | 0.658 | 0.167 |
| 4 | 8 | 4.822 | 0.170 | 0.803 | 0.998 | 0.684 |
| 4 | 16 | 5.314 | 0.124 | 0.712 | 0.992 | 0.569 |
| 4 | 32 | 5.511 | 0.119 | 0.634 | 0.982 | 0.513 |
| 4 | 64 | 5.765 | 0.090 | 0.549 | 0.967 | 0.430 |
| 4 | 128 | 5.524 | 0.125 | 0.661 | 0.982 | 0.515 |
| 5 | 2 | 7.327 | 0.031 | 0.235 | 0.662 | 0.166 |
| 5 | 4 | 7.211 | 0.039 | 0.258 | 0.683 | 0.178 |
| 5 | 8 | 4.931 | 0.165 | 0.787 | 0.997 | 0.659 |
| 5 | 16 | 4.901 | 0.177 | 0.788 | 0.997 | 0.669 |
| 5 | 32 | 4.874 | 0.163 | 0.810 | 0.999 | 0.684 |
| 5 | 64 | 5.673 | 0.110 | 0.596 | 0.972 | 0.468 |
| 5 | 128 | 5.066 | 0.148 | 0.754 | 0.995 | 0.635 |
| 6 | 2 | 7.327 | 0.031 | 0.235 | 0.662 | 0.166 |
| 6 | 4 | 5.189 | 0.132 | 0.714 | 0.994 | 0.557 |
| 6 | 8 | 4.807 | 0.168 | 0.826 | 0.999 | 0.690 |
| 6 | 16 | 4.740 | 0.150 | 0.830 | 0.999 | 0.705 |
| 6 | 32 | 5.163 | 0.127 | 0.755 | 0.998 | 0.599 |
| 6 | 64 | 5.502 | 0.122 | 0.689 | 0.989 | 0.538 |
| 6 | 128 | 5.092 | 0.151 | 0.778 | 0.998 | 0.637 |

*Table B.13: Model Metrics for N = 6 (Distinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 52 | 58 | 213.9 | 16,064 |
| 1 | 4 | 52 | 46 | 222.5 | 25,300 |
| 1 | 8 | 52 | 46 | 265.8 | 43,796 |
| 1 | 16 | 52 | 38 | 49.5 | 80,884 |
| 1 | 32 | 52 | 29 | 40.8 | 155,444 |
| 1 | 64 | 52 | 29 | 62.9 | 306,100 |
| 1 | 128 | 52 | 27 | 102.2 | 613,556 |
| 2 | 2 | 52 | 48 | 190.2 | 16,070 |
| 2 | 4 | 52 | 91 | 373.5 | 25,320 |
| 2 | 8 | 52 | 34 | 196.1 | 43,868 |
| 2 | 16 | 52 | 30 | 50.8 | 81,156 |
| 2 | 32 | 52 | 23 | 47.3 | 156,500 |
| 2 | 64 | 52 | 20 | 55.0 | 310,260 |
| 2 | 128 | 52 | 24 | 123.6 | 630,068 |
| 3 | 2 | 52 | 57 | 243.7 | 16,076 |
| 3 | 4 | 52 | 54 | 240.7 | 25,340 |
| 3 | 8 | 52 | 38 | 225.6 | 43,940 |
| 3 | 16 | 52 | 29 | 65.5 | 81,428 |
| 3 | 32 | 52 | 26 | 60.2 | 157,556 |
| 3 | 64 | 52 | 22 | 66.5 | 314,420 |
| 3 | 128 | 52 | 20 | 124.2 | 646,580 |
| 4 | 2 | 52 | 56 | 230.6 | 16,082 |
| 4 | 4 | 52 | 271 | 1416.8 | 25,360 |
| 4 | 8 | 52 | 43 | 276.6 | 44,012 |
| 4 | 16 | 52 | 30 | 67.1 | 81,700 |
| 4 | 32 | 52 | 27 | 70.1 | 158,612 |
| 4 | 64 | 52 | 27 | 100.4 | 318,580 |
| 4 | 128 | 52 | 20 | 148.7 | 663,092 |
| 5 | 2 | 52 | 52 | 272.0 | 16,088 |
| 5 | 4 | 52 | 165 | 988.3 | 25,380 |
| 5 | 8 | 52 | 65 | 454.6 | 44,084 |
| 5 | 16 | 52 | 141 | 228.2 | 81,972 |
| 5 | 32 | 52 | 123 | 256.8 | 159,668 |
| 5 | 64 | 52 | 26 | 86.3 | 322,740 |
| 5 | 128 | 52 | 30 | 207.0 | 679,604 |

*Continued on next page*

*Table B.13 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 6 | 2 | 52 | 50 | 280.8 | 16,094 |
| 6 | 4 | 52 | 226 | 1378.1 | 25,400 |
| 6 | 8 | 52 | 35 | 286.7 | 44,156 |
| 6 | 16 | 52 | 33 | 70.9 | 82,244 |
| 6 | 32 | 52 | 25 | 62.3 | 160,724 |
| 6 | 64 | 52 | 31 | 114.5 | 326,900 |
| 6 | 128 | 52 | 27 | 203.7 | 696,116 |

*Table B.14: Training Metrics for N = 6 (Distinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.322 | 0.039 | 0.254 | 0.675 | 0.183 |
| 1 | 4 | 5.378 | 0.180 | 0.629 | 0.978 | 0.580 |
| 1 | 8 | 5.192 | 0.237 | 0.675 | 0.982 | 0.654 |
| 1 | 16 | 5.249 | 0.239 | 0.657 | 0.973 | 0.650 |
| 1 | 32 | 5.279 | 0.230 | 0.625 | 0.978 | 0.625 |
| 1 | 64 | 5.070 | 0.289 | 0.687 | 0.981 | 0.695 |
| 1 | 128 | 5.363 | 0.235 | 0.627 | 0.964 | 0.627 |
| 2 | 2 | 7.322 | 0.039 | 0.254 | 0.675 | 0.183 |
| 2 | 4 | 5.184 | 0.234 | 0.685 | 0.979 | 0.662 |
| 2 | 8 | 5.326 | 0.197 | 0.636 | 0.980 | 0.595 |
| 2 | 16 | 5.237 | 0.229 | 0.644 | 0.981 | 0.639 |
| 2 | 32 | 5.289 | 0.227 | 0.650 | 0.974 | 0.637 |
| 2 | 64 | 5.233 | 0.218 | 0.662 | 0.983 | 0.633 |
| 2 | 128 | 5.196 | 0.248 | 0.646 | 0.974 | 0.661 |
| 3 | 2 | 7.322 | 0.039 | 0.254 | 0.675 | 0.183 |
| 3 | 4 | 5.536 | 0.145 | 0.578 | 0.978 | 0.500 |
| 3 | 8 | 5.326 | 0.194 | 0.635 | 0.980 | 0.593 |
| 3 | 16 | 5.178 | 0.251 | 0.680 | 0.975 | 0.670 |
| 3 | 32 | 5.361 | 0.210 | 0.631 | 0.972 | 0.609 |
| 3 | 64 | 5.281 | 0.212 | 0.648 | 0.975 | 0.634 |
| 3 | 128 | 5.215 | 0.227 | 0.662 | 0.979 | 0.656 |
| 4 | 2 | 7.322 | 0.039 | 0.254 | 0.675 | 0.183 |
| 4 | 4 | 4.253 | 0.391 | 0.876 | 0.998 | 0.865 |
| 4 | 8 | 5.355 | 0.187 | 0.649 | 0.979 | 0.583 |

*Continued on next page*

*Table B.14 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 4 | 16 | 5.284 | 0.198 | 0.634 | 0.984 | 0.605 |
| 4 | 32 | 5.543 | 0.165 | 0.560 | 0.966 | 0.525 |
| 4 | 64 | 5.106 | 0.267 | 0.668 | 0.980 | 0.684 |
| 4 | 128 | 5.317 | 0.189 | 0.626 | 0.982 | 0.592 |
| 5 | 2 | 7.322 | 0.039 | 0.254 | 0.675 | 0.183 |
| 5 | 4 | 4.314 | 0.374 | 0.856 | 0.998 | 0.850 |
| 5 | 8 | 5.282 | 0.184 | 0.664 | 0.979 | 0.609 |
| 5 | 16 | 4.166 | 0.349 | 0.879 | 1.000 | 0.858 |
| 5 | 32 | 4.716 | 0.230 | 0.761 | 0.998 | 0.709 |
| 5 | 64 | 5.139 | 0.254 | 0.682 | 0.975 | 0.677 |
| 5 | 128 | 5.108 | 0.217 | 0.690 | 0.986 | 0.647 |
| 6 | 2 | 7.322 | 0.039 | 0.254 | 0.675 | 0.183 |
| 6 | 4 | 4.628 | 0.305 | 0.768 | 0.995 | 0.764 |
| 6 | 8 | 5.318 | 0.172 | 0.639 | 0.982 | 0.576 |
| 6 | 16 | 5.308 | 0.178 | 0.657 | 0.981 | 0.597 |
| 6 | 32 | 5.256 | 0.184 | 0.667 | 0.984 | 0.608 |
| 6 | 64 | 5.247 | 0.185 | 0.637 | 0.985 | 0.590 |
| 6 | 128 | 5.502 | 0.171 | 0.597 | 0.968 | 0.553 |

*Table B.15: Validation Metrics for N = 6 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.322 | 0.046 | 0.264 | 0.664 | 0.204 |
| 1 | 4 | 5.590 | 0.110 | 0.594 | 0.977 | 0.452 |
| 1 | 8 | 5.617 | 0.111 | 0.584 | 0.972 | 0.447 |
| 1 | 16 | 5.683 | 0.099 | 0.574 | 0.972 | 0.440 |
| 1 | 32 | 5.681 | 0.082 | 0.547 | 0.975 | 0.402 |
| 1 | 64 | 5.687 | 0.086 | 0.566 | 0.973 | 0.425 |
| 1 | 128 | 5.854 | 0.083 | 0.553 | 0.966 | 0.420 |
| 2 | 2 | 7.321 | 0.046 | 0.264 | 0.664 | 0.204 |
| 2 | 4 | 5.670 | 0.099 | 0.580 | 0.966 | 0.437 |
| 2 | 8 | 5.629 | 0.099 | 0.569 | 0.977 | 0.441 |
| 2 | 16 | 5.607 | 0.104 | 0.573 | 0.977 | 0.443 |
| 2 | 32 | 5.671 | 0.096 | 0.578 | 0.973 | 0.427 |
| 2 | 64 | 5.581 | 0.093 | 0.590 | 0.973 | 0.446 |

*Continued on next page*

*Table B.15 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 2 | 128 | 5.752 | 0.106 | 0.553 | 0.973 | 0.411 |
| 3 | 2 | 7.322 | 0.046 | 0.264 | 0.664 | 0.204 |
| 3 | 4 | 5.667 | 0.109 | 0.573 | 0.976 | 0.453 |
| 3 | 8 | 5.608 | 0.106 | 0.590 | 0.980 | 0.448 |
| 3 | 16 | 5.660 | 0.094 | 0.570 | 0.976 | 0.430 |
| 3 | 32 | 5.745 | 0.093 | 0.573 | 0.968 | 0.440 |
| 3 | 64 | 5.705 | 0.097 | 0.575 | 0.971 | 0.445 |
| 3 | 128 | 5.565 | 0.119 | 0.607 | 0.979 | 0.480 |
| 4 | 2 | 7.322 | 0.046 | 0.264 | 0.664 | 0.204 |
| 4 | 4 | 5.376 | 0.121 | 0.687 | 0.994 | 0.550 |
| 4 | 8 | 5.635 | 0.109 | 0.584 | 0.971 | 0.457 |
| 4 | 16 | 5.552 | 0.097 | 0.585 | 0.981 | 0.458 |
| 4 | 32 | 5.875 | 0.073 | 0.499 | 0.966 | 0.375 |
| 4 | 64 | 5.737 | 0.086 | 0.550 | 0.975 | 0.416 |
| 4 | 128 | 5.627 | 0.107 | 0.576 | 0.977 | 0.462 |
| 5 | 2 | 7.321 | 0.046 | 0.264 | 0.664 | 0.204 |
| 5 | 4 | 5.279 | 0.126 | 0.700 | 0.995 | 0.558 |
| 5 | 8 | 5.594 | 0.109 | 0.602 | 0.979 | 0.467 |
| 5 | 16 | 5.145 | 0.138 | 0.765 | 0.998 | 0.614 |
| 5 | 32 | 5.620 | 0.107 | 0.655 | 0.990 | 0.513 |
| 5 | 64 | 5.788 | 0.086 | 0.575 | 0.968 | 0.430 |
| 5 | 128 | 5.536 | 0.096 | 0.604 | 0.981 | 0.468 |
| 6 | 2 | 7.321 | 0.046 | 0.264 | 0.664 | 0.204 |
| 6 | 4 | 5.556 | 0.106 | 0.639 | 0.990 | 0.492 |
| 6 | 8 | 5.524 | 0.116 | 0.603 | 0.981 | 0.478 |
| 6 | 16 | 5.599 | 0.104 | 0.600 | 0.978 | 0.461 |
| 6 | 32 | 5.567 | 0.093 | 0.623 | 0.981 | 0.470 |
| 6 | 64 | 5.654 | 0.112 | 0.594 | 0.977 | 0.451 |
| 6 | 128 | 5.895 | 0.087 | 0.540 | 0.964 | 0.400 |

*Table B.16: Model Metrics for N = 7 (Distinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 51 | 52 | 21.3 | 19,263 |
| 1 | 4 | 51 | 73 | 28.7 | 31,827 |

*Continued on next page*

*Table B.16 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 8 | 51 | 39 | 18.1 | 56,979 |
| 1 | 16 | 51 | 39 | 23.2 | 107,379 |
| 1 | 32 | 51 | 31 | 28.2 | 208,563 |
| 1 | 64 | 51 | 28 | 49.9 | 412,467 |
| 1 | 128 | 51 | 21 | 76.1 | 826,419 |
| 2 | 2 | 51 | 11 | 5.9 | 19,269 |
| 2 | 4 | 51 | 75 | 33.6 | 31,847 |
| 2 | 8 | 51 | 39 | 20.9 | 57,051 |
| 2 | 16 | 51 | 34 | 25.0 | 107,651 |
| 2 | 32 | 51 | 27 | 29.4 | 209,619 |
| 2 | 64 | 51 | 24 | 57.3 | 416,627 |
| 2 | 128 | 51 | 20 | 93.6 | 842,931 |
| 3 | 2 | 51 | 71 | 34.0 | 19,275 |
| 3 | 4 | 51 | 56 | 29.2 | 31,867 |
| 3 | 8 | 51 | 39 | 24.3 | 57,123 |
| 3 | 16 | 51 | 38 | 33.0 | 107,923 |
| 3 | 32 | 51 | 28 | 41.2 | 210,675 |
| 3 | 64 | 51 | 27 | 84.5 | 420,787 |
| 3 | 128 | 51 | 22 | 136.5 | 859,443 |
| 4 | 2 | 51 | 18 | 11.3 | 19,281 |
| 4 | 4 | 51 | 16 | 11.2 | 31,887 |
| 4 | 8 | 51 | 47 | 33.0 | 57,195 |
| 4 | 16 | 51 | 31 | 28.4 | 108,195 |
| 4 | 32 | 51 | 28 | 38.7 | 211,731 |
| 4 | 64 | 51 | 31 | 97.3 | 424,947 |
| 4 | 128 | 51 | 19 | 132.7 | 875,955 |
| 5 | 2 | 51 | 67 | 37.8 | 19,287 |
| 5 | 4 | 51 | 48 | 30.3 | 31,907 |
| 5 | 8 | 51 | 45 | 33.5 | 57,267 |
| 5 | 16 | 51 | 30 | 34.5 | 108,467 |
| 5 | 32 | 51 | 24 | 43.2 | 212,787 |
| 5 | 64 | 51 | 26 | 101.4 | 429,107 |
| 5 | 128 | 51 | 31 | 261.3 | 892,467 |
| 6 | 2 | 51 | 88 | 53.9 | 19,293 |
| 6 | 4 | 51 | 14 | 11.8 | 31,927 |
| 6 | 8 | 51 | 49 | 40.3 | 57,339 |

*Continued on next page*

*Table B.16 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 6 | 16 | 51 | 34 | 44.1 | 108,739 |
| 6 | 32 | 51 | 28 | 56.9 | 213,843 |
| 6 | 64 | 51 | 28 | 129.1 | 433,267 |
| 6 | 128 | 51 | 24 | 229.3 | 908,979 |

*Table B.17: Training Metrics for N = 7 (Distinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 6.672 | 0.108 | 0.356 | 0.766 | 0.353 |
| 1 | 4 | 5.274 | 0.225 | 0.642 | 0.973 | 0.630 |
| 1 | 8 | 4.968 | 0.310 | 0.710 | 0.985 | 0.723 |
| 1 | 16 | 4.885 | 0.330 | 0.724 | 0.987 | 0.742 |
| 1 | 32 | 5.206 | 0.264 | 0.633 | 0.975 | 0.648 |
| 1 | 64 | 4.878 | 0.328 | 0.726 | 0.988 | 0.749 |
| 1 | 128 | 4.978 | 0.286 | 0.709 | 0.989 | 0.719 |
| 2 | 2 | 7.311 | 0.039 | 0.261 | 0.667 | 0.189 |
| 2 | 4 | 7.306 | 0.040 | 0.257 | 0.664 | 0.190 |
| 2 | 8 | 5.113 | 0.230 | 0.688 | 0.990 | 0.650 |
| 2 | 16 | 5.157 | 0.225 | 0.656 | 0.986 | 0.636 |
| 2 | 32 | 4.972 | 0.303 | 0.706 | 0.987 | 0.721 |
| 2 | 64 | 4.996 | 0.277 | 0.713 | 0.986 | 0.716 |
| 2 | 128 | 5.438 | 0.229 | 0.594 | 0.962 | 0.611 |
| 3 | 2 | 7.306 | 0.040 | 0.257 | 0.664 | 0.190 |
| 3 | 4 | 5.157 | 0.226 | 0.665 | 0.986 | 0.646 |
| 3 | 8 | 5.235 | 0.193 | 0.658 | 0.988 | 0.595 |
| 3 | 16 | 5.049 | 0.250 | 0.693 | 0.987 | 0.681 |
| 3 | 32 | 4.971 | 0.298 | 0.711 | 0.983 | 0.716 |
| 3 | 64 | 5.060 | 0.255 | 0.697 | 0.985 | 0.692 |
| 3 | 128 | 5.004 | 0.262 | 0.697 | 0.988 | 0.702 |
| 4 | 2 | 7.307 | 0.040 | 0.257 | 0.664 | 0.190 |
| 4 | 4 | 7.308 | 0.038 | 0.258 | 0.663 | 0.190 |
| 4 | 8 | 5.122 | 0.230 | 0.678 | 0.987 | 0.645 |
| 4 | 16 | 5.232 | 0.225 | 0.627 | 0.983 | 0.614 |
| 4 | 32 | 5.112 | 0.237 | 0.688 | 0.985 | 0.671 |
| 4 | 64 | 5.197 | 0.220 | 0.667 | 0.982 | 0.638 |

*Continued on next page*

*Table B.17 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 4 | 128 | 5.072 | 0.242 | 0.683 | 0.985 | 0.687 |
| 5 | 2 | 7.306 | 0.040 | 0.257 | 0.664 | 0.190 |
| 5 | 4 | 5.200 | 0.213 | 0.662 | 0.987 | 0.617 |
| 5 | 8 | 4.993 | 0.258 | 0.712 | 0.987 | 0.693 |
| 5 | 16 | 5.176 | 0.209 | 0.666 | 0.989 | 0.623 |
| 5 | 32 | 5.101 | 0.228 | 0.692 | 0.987 | 0.662 |
| 5 | 64 | 5.027 | 0.249 | 0.709 | 0.989 | 0.683 |
| 5 | 128 | 4.867 | 0.285 | 0.736 | 0.987 | 0.728 |
| 6 | 2 | 7.306 | 0.040 | 0.257 | 0.664 | 0.190 |
| 6 | 4 | 7.306 | 0.037 | 0.262 | 0.666 | 0.193 |
| 6 | 8 | 4.684 | 0.357 | 0.776 | 0.986 | 0.793 |
| 6 | 16 | 5.124 | 0.208 | 0.686 | 0.987 | 0.634 |
| 6 | 32 | 5.056 | 0.240 | 0.712 | 0.988 | 0.678 |
| 6 | 64 | 4.957 | 0.294 | 0.712 | 0.977 | 0.721 |
| 6 | 128 | 4.975 | 0.237 | 0.717 | 0.988 | 0.688 |

*Table B.18: Validation Metrics for N = 7 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 6.858 | 0.046 | 0.307 | 0.762 | 0.213 |
| 1 | 4 | 5.760 | 0.093 | 0.528 | 0.966 | 0.417 |
| 1 | 8 | 5.656 | 0.109 | 0.549 | 0.980 | 0.442 |
| 1 | 16 | 5.558 | 0.122 | 0.591 | 0.981 | 0.468 |
| 1 | 32 | 5.843 | 0.091 | 0.519 | 0.975 | 0.401 |
| 1 | 64 | 5.591 | 0.119 | 0.593 | 0.982 | 0.463 |
| 1 | 128 | 5.540 | 0.112 | 0.610 | 0.984 | 0.479 |
| 2 | 2 | 7.293 | 0.036 | 0.262 | 0.686 | 0.187 |
| 2 | 4 | 7.283 | 0.044 | 0.254 | 0.669 | 0.195 |
| 2 | 8 | 5.466 | 0.114 | 0.608 | 0.986 | 0.473 |
| 2 | 16 | 5.512 | 0.115 | 0.588 | 0.984 | 0.464 |
| 2 | 32 | 5.541 | 0.121 | 0.601 | 0.983 | 0.466 |
| 2 | 64 | 5.616 | 0.120 | 0.613 | 0.979 | 0.476 |
| 2 | 128 | 6.033 | 0.091 | 0.522 | 0.957 | 0.401 |
| 3 | 2 | 7.283 | 0.044 | 0.254 | 0.669 | 0.195 |
| 3 | 4 | 5.581 | 0.115 | 0.588 | 0.983 | 0.460 |

*Continued on next page*

*Table B.18 – continued from previous page*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 3 | 8 | 5.479 | 0.118 | 0.608 | 0.984 | 0.484 |
| 3 | 16 | 5.494 | 0.129 | 0.593 | 0.986 | 0.493 |
| 3 | 32 | 5.615 | 0.107 | 0.600 | 0.983 | 0.464 |
| 3 | 64 | 5.553 | 0.114 | 0.604 | 0.984 | 0.475 |
| 3 | 128 | 5.608 | 0.107 | 0.572 | 0.987 | 0.456 |
| 4 | 2 | 7.284 | 0.044 | 0.254 | 0.669 | 0.195 |
| 4 | 4 | 7.283 | 0.038 | 0.260 | 0.684 | 0.185 |
| 4 | 8 | 5.537 | 0.105 | 0.592 | 0.985 | 0.469 |
| 4 | 16 | 5.679 | 0.098 | 0.552 | 0.979 | 0.428 |
| 4 | 32 | 5.570 | 0.107 | 0.604 | 0.984 | 0.478 |
| 4 | 64 | 5.606 | 0.118 | 0.586 | 0.982 | 0.468 |
| 4 | 128 | 5.623 | 0.104 | 0.575 | 0.976 | 0.455 |
| 5 | 2 | 7.283 | 0.044 | 0.254 | 0.669 | 0.195 |
| 5 | 4 | 5.531 | 0.113 | 0.589 | 0.982 | 0.466 |
| 5 | 8 | 5.600 | 0.105 | 0.592 | 0.982 | 0.476 |
| 5 | 16 | 5.519 | 0.097 | 0.594 | 0.986 | 0.468 |
| 5 | 32 | 5.493 | 0.113 | 0.605 | 0.989 | 0.473 |
| 5 | 64 | 5.541 | 0.102 | 0.607 | 0.984 | 0.472 |
| 5 | 128 | 5.662 | 0.106 | 0.589 | 0.986 | 0.467 |
| 6 | 2 | 7.283 | 0.044 | 0.254 | 0.669 | 0.195 |
| 6 | 4 | 7.285 | 0.040 | 0.260 | 0.687 | 0.192 |
| 6 | 8 | 5.889 | 0.094 | 0.541 | 0.969 | 0.423 |
| 6 | 16 | 5.435 | 0.113 | 0.621 | 0.989 | 0.479 |
| 6 | 32 | 5.512 | 0.108 | 0.630 | 0.987 | 0.498 |
| 6 | 64 | 5.777 | 0.098 | 0.578 | 0.977 | 0.453 |
| 6 | 128 | 5.574 | 0.102 | 0.602 | 0.983 | 0.452 |

*Table B.19: Model Metrics for N = 8 (Distinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 50 | 13 | 11.0 | 22,974 |
| 1 | 4 | 50 | 51 | 37.8 | 39,378 |
| 1 | 8 | 50 | 47 | 38.4 | 72,210 |
| 1 | 16 | 50 | 39 | 28.7 | 137,970 |
| 1 | 32 | 50 | 27 | 34.5 | 269,874 |

*Continued on next page*

*Table B.19 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 64 | 50 | 26 | 60.2 | 535,218 |
| 1 | 128 | 50 | 25 | 124.0 | 1,072,050 |
| 2 | 2 | 50 | 45 | 32.3 | 22,980 |
| 2 | 4 | 50 | 77 | 62.3 | 39,398 |
| 2 | 8 | 50 | 39 | 37.8 | 72,282 |
| 2 | 16 | 50 | 33 | 31.0 | 138,242 |
| 2 | 32 | 50 | 28 | 44.7 | 270,930 |
| 2 | 64 | 50 | 20 | 62.0 | 539,378 |
| 2 | 128 | 50 | 19 | 121.6 | 1,088,562 |
| 3 | 2 | 50 | 46 | 37.7 | 22,986 |
| 3 | 4 | 50 | 16 | 19.3 | 39,418 |
| 3 | 8 | 50 | 54 | 61.8 | 72,354 |
| 3 | 16 | 50 | 36 | 48.8 | 138,514 |
| 3 | 32 | 50 | 31 | 69.8 | 271,986 |
| 3 | 64 | 50 | 32 | 129.9 | 543,538 |
| 3 | 128 | 50 | 19 | 161.0 | 1,105,074 |
| 4 | 2 | 50 | 43 | 40.2 | 22,992 |
| 4 | 4 | 50 | 30 | 37.5 | 39,438 |
| 4 | 8 | 50 | 51 | 67.9 | 72,426 |
| 4 | 16 | 50 | 47 | 84.9 | 138,786 |
| 4 | 32 | 50 | 27 | 72.7 | 273,042 |
| 4 | 64 | 50 | 26 | 124.9 | 547,698 |
| 4 | 128 | 50 | 21 | 188.5 | 1,121,586 |
| 5 | 2 | 50 | 32 | 35.3 | 22,998 |
| 5 | 4 | 50 | 89 | 120.5 | 39,458 |
| 5 | 8 | 50 | 38 | 60.5 | 72,498 |
| 5 | 16 | 50 | 37 | 68.5 | 139,058 |
| 5 | 32 | 50 | 23 | 67.1 | 274,098 |
| 5 | 64 | 50 | 25 | 127.6 | 551,858 |
| 5 | 128 | 50 | 21 | 226.0 | 1,138,098 |
| 6 | 2 | 50 | 46 | 54.2 | 23,004 |
| 6 | 4 | 50 | 17 | 33.9 | 39,478 |
| 6 | 8 | 50 | 40 | 78.8 | 72,570 |
| 6 | 16 | 50 | 32 | 73.9 | 139,330 |
| 6 | 32 | 50 | 28 | 97.8 | 275,154 |
| 6 | 64 | 50 | 23 | 136.8 | 556,018 |

*Continued on next page*

*Table B.19 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 6 | 128 | 50 | 23 | 270.6 | 1,154,610 |

*Table B.20: Training Metrics for N = 8 (Distinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.066 | 0.093 | 0.301 | 0.675 | 0.309 |
| 1 | 4 | 5.100 | 0.284 | 0.676 | 0.978 | 0.685 |
| 1 | 8 | 4.844 | 0.344 | 0.722 | 0.991 | 0.750 |
| 1 | 16 | 4.611 | 0.432 | 0.779 | 0.992 | 0.827 |
| 1 | 32 | 4.832 | 0.336 | 0.737 | 0.992 | 0.756 |
| 1 | 64 | 4.663 | 0.406 | 0.765 | 0.993 | 0.807 |
| 1 | 128 | 4.787 | 0.365 | 0.737 | 0.988 | 0.774 |
| 2 | 2 | 7.255 | 0.040 | 0.266 | 0.683 | 0.194 |
| 2 | 4 | 5.869 | 0.222 | 0.532 | 0.854 | 0.574 |
| 2 | 8 | 4.995 | 0.281 | 0.702 | 0.989 | 0.704 |
| 2 | 16 | 4.766 | 0.386 | 0.744 | 0.982 | 0.785 |
| 2 | 32 | 4.881 | 0.307 | 0.733 | 0.990 | 0.733 |
| 2 | 64 | 4.837 | 0.320 | 0.740 | 0.990 | 0.751 |
| 2 | 128 | 4.982 | 0.321 | 0.701 | 0.981 | 0.722 |
| 3 | 2 | 7.255 | 0.040 | 0.266 | 0.683 | 0.194 |
| 3 | 4 | 7.240 | 0.050 | 0.265 | 0.678 | 0.211 |
| 3 | 8 | 5.183 | 0.286 | 0.664 | 0.969 | 0.669 |
| 3 | 16 | 4.969 | 0.271 | 0.701 | 0.988 | 0.695 |
| 3 | 32 | 4.760 | 0.355 | 0.746 | 0.991 | 0.774 |
| 3 | 64 | 4.987 | 0.288 | 0.688 | 0.988 | 0.698 |
| 3 | 128 | 4.498 | 0.487 | 0.796 | 0.978 | 0.862 |
| 4 | 2 | 7.255 | 0.040 | 0.266 | 0.683 | 0.194 |
| 4 | 4 | 6.522 | 0.166 | 0.415 | 0.720 | 0.471 |
| 4 | 8 | 4.954 | 0.248 | 0.714 | 0.990 | 0.688 |
| 4 | 16 | 5.097 | 0.229 | 0.697 | 0.988 | 0.648 |
| 4 | 32 | 5.404 | 0.202 | 0.616 | 0.977 | 0.583 |
| 4 | 64 | 4.795 | 0.325 | 0.757 | 0.990 | 0.756 |
| 4 | 128 | 4.705 | 0.331 | 0.771 | 0.990 | 0.791 |
| 5 | 2 | 7.255 | 0.039 | 0.258 | 0.673 | 0.192 |
| 5 | 4 | 5.195 | 0.190 | 0.667 | 0.989 | 0.612 |

*Continued on next page*

*Table B.20 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 5 | 8 | 5.043 | 0.218 | 0.694 | 0.992 | 0.654 |
| 5 | 16 | 5.096 | 0.241 | 0.658 | 0.984 | 0.658 |
| 5 | 32 | 4.954 | 0.265 | 0.727 | 0.988 | 0.696 |
| 5 | 64 | 4.868 | 0.300 | 0.731 | 0.985 | 0.734 |
| 5 | 128 | 4.956 | 0.267 | 0.718 | 0.992 | 0.709 |
| 6 | 2 | 7.255 | 0.040 | 0.266 | 0.683 | 0.194 |
| 6 | 4 | 7.256 | 0.040 | 0.266 | 0.683 | 0.194 |
| 6 | 8 | 5.030 | 0.273 | 0.698 | 0.981 | 0.696 |
| 6 | 16 | 4.957 | 0.261 | 0.716 | 0.984 | 0.702 |
| 6 | 32 | 4.840 | 0.296 | 0.730 | 0.993 | 0.733 |
| 6 | 64 | 4.856 | 0.282 | 0.745 | 0.991 | 0.734 |
| 6 | 128 | 4.945 | 0.266 | 0.721 | 0.989 | 0.699 |

*Table B.21: Validation Metrics for N = 8 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.326 | 0.036 | 0.263 | 0.636 | 0.166 |
| 1 | 4 | 5.668 | 0.109 | 0.557 | 0.978 | 0.425 |
| 1 | 8 | 5.605 | 0.104 | 0.570 | 0.984 | 0.450 |
| 1 | 16 | 5.727 | 0.100 | 0.568 | 0.980 | 0.426 |
| 1 | 32 | 5.497 | 0.107 | 0.607 | 0.988 | 0.468 |
| 1 | 64 | 5.626 | 0.106 | 0.597 | 0.987 | 0.462 |
| 1 | 128 | 5.764 | 0.102 | 0.580 | 0.980 | 0.447 |
| 2 | 2 | 7.286 | 0.039 | 0.253 | 0.667 | 0.182 |
| 2 | 4 | 6.723 | 0.057 | 0.338 | 0.817 | 0.259 |
| 2 | 8 | 5.523 | 0.107 | 0.608 | 0.985 | 0.472 |
| 2 | 16 | 5.841 | 0.101 | 0.569 | 0.974 | 0.441 |
| 2 | 32 | 5.504 | 0.109 | 0.621 | 0.989 | 0.488 |
| 2 | 64 | 5.498 | 0.114 | 0.621 | 0.987 | 0.475 |
| 2 | 128 | 5.900 | 0.108 | 0.564 | 0.971 | 0.428 |
| 3 | 2 | 7.286 | 0.039 | 0.253 | 0.667 | 0.182 |
| 3 | 4 | 7.286 | 0.036 | 0.245 | 0.659 | 0.179 |
| 3 | 8 | 6.020 | 0.087 | 0.541 | 0.969 | 0.415 |
| 3 | 16 | 5.388 | 0.105 | 0.627 | 0.992 | 0.494 |
| 3 | 32 | 5.557 | 0.115 | 0.607 | 0.990 | 0.469 |

*Continued on next page*

*Table B.21 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 3 | 64 | 5.605 | 0.113 | 0.589 | 0.990 | 0.443 |
| 3 | 128 | 6.108 | 0.090 | 0.544 | 0.972 | 0.426 |
| 4 | 2 | 7.286 | 0.039 | 0.253 | 0.667 | 0.182 |
| 4 | 4 | 7.407 | 0.042 | 0.250 | 0.651 | 0.180 |
| 4 | 8 | 5.427 | 0.127 | 0.620 | 0.986 | 0.500 |
| 4 | 16 | 5.403 | 0.110 | 0.631 | 0.987 | 0.520 |
| 4 | 32 | 5.717 | 0.116 | 0.570 | 0.982 | 0.455 |
| 4 | 64 | 5.469 | 0.111 | 0.635 | 0.990 | 0.496 |
| 4 | 128 | 5.581 | 0.111 | 0.612 | 0.988 | 0.488 |
| 5 | 2 | 7.286 | 0.038 | 0.246 | 0.653 | 0.182 |
| 5 | 4 | 5.435 | 0.125 | 0.628 | 0.985 | 0.492 |
| 5 | 8 | 5.327 | 0.135 | 0.642 | 0.991 | 0.501 |
| 5 | 16 | 5.652 | 0.100 | 0.561 | 0.986 | 0.437 |
| 5 | 32 | 5.386 | 0.118 | 0.654 | 0.988 | 0.493 |
| 5 | 64 | 5.576 | 0.112 | 0.623 | 0.985 | 0.475 |
| 5 | 128 | 5.445 | 0.121 | 0.615 | 0.988 | 0.486 |
| 6 | 2 | 7.286 | 0.039 | 0.253 | 0.667 | 0.182 |
| 6 | 4 | 7.289 | 0.039 | 0.253 | 0.667 | 0.182 |
| 6 | 8 | 5.765 | 0.105 | 0.565 | 0.978 | 0.428 |
| 6 | 16 | 5.595 | 0.106 | 0.600 | 0.985 | 0.471 |
| 6 | 32 | 5.420 | 0.112 | 0.619 | 0.988 | 0.490 |
| 6 | 64 | 5.404 | 0.123 | 0.640 | 0.990 | 0.517 |
| 6 | 128 | 5.468 | 0.129 | 0.636 | 0.989 | 0.492 |

*Table B.22: Model Metrics for N = 9 (Distinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 50 | 15 | 29.1 | 27,326 |
| 1 | 4 | 50 | 45 | 83.0 | 48,082 |
| 1 | 8 | 50 | 12 | 29.9 | 89,618 |
| 1 | 16 | 50 | 34 | 35.0 | 172,786 |
| 1 | 32 | 50 | 29 | 51.3 | 339,506 |
| 1 | 64 | 50 | 26 | 79.7 | 674,482 |
| 1 | 128 | 50 | 23 | 139.7 | 1,350,578 |
| 2 | 2 | 50 | 62 | 109.0 | 27,332 |

*Continued on next page*

*Table B.22 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 2 | 4 | 50 | 61 | 124.1 | 48,102 |
| 2 | 8 | 50 | 43 | 101.4 | 89,690 |
| 2 | 16 | 50 | 32 | 41.4 | 173,058 |
| 2 | 32 | 50 | 28 | 61.8 | 340,562 |
| 2 | 64 | 50 | 25 | 95.2 | 678,642 |
| 2 | 128 | 50 | 23 | 176.5 | 1,367,090 |
| 3 | 2 | 50 | 21 | 45.4 | 27,338 |
| 3 | 4 | 50 | 15 | 40.5 | 48,122 |
| 3 | 8 | 50 | 50 | 130.6 | 89,762 |
| 3 | 16 | 50 | 38 | 89.8 | 173,330 |
| 3 | 32 | 50 | 33 | 96.3 | 341,618 |
| 3 | 64 | 50 | 21 | 104.7 | 682,802 |
| 3 | 128 | 50 | 20 | 231.7 | 1,383,602 |
| 4 | 2 | 50 | 83 | 165.6 | 27,344 |
| 4 | 4 | 50 | 75 | 191.6 | 48,142 |
| 4 | 8 | 50 | 43 | 128.1 | 89,834 |
| 4 | 16 | 50 | 31 | 92.7 | 173,602 |
| 4 | 32 | 50 | 31 | 114.7 | 342,674 |
| 4 | 64 | 50 | 24 | 149.4 | 686,962 |
| 4 | 128 | 50 | 18 | 229.5 | 1,400,114 |
| 5 | 2 | 50 | 76 | 168.5 | 27,350 |
| 5 | 4 | 50 | 16 | 52.8 | 48,162 |
| 5 | 8 | 50 | 49 | 165.2 | 89,906 |
| 5 | 16 | 50 | 32 | 107.0 | 173,874 |
| 5 | 32 | 50 | 25 | 116.4 | 343,730 |
| 5 | 64 | 50 | 21 | 158.3 | 691,122 |
| 5 | 128 | 50 | 24 | 365.3 | 1,416,626 |
| 6 | 2 | 50 | 82 | 200.3 | 27,356 |
| 6 | 4 | 50 | 16 | 57.6 | 48,182 |
| 6 | 8 | 50 | 69 | 251.4 | 89,978 |
| 6 | 16 | 50 | 31 | 137.7 | 174,146 |
| 6 | 32 | 50 | 33 | 177.4 | 344,786 |
| 6 | 64 | 50 | 25 | 220.6 | 695,282 |
| 6 | 128 | 50 | 25 | 412.7 | 1,433,138 |

*Table B.23: Training Metrics for N = 9 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 6.912 | 0.103 | 0.326 | 0.694 | 0.349 |
| 1 | 4 | 4.665 | 0.465 | 0.789 | 0.968 | 0.856 |
| 1 | 8 | 6.812 | 0.120 | 0.343 | 0.713 | 0.376 |
| 1 | 16 | 4.813 | 0.349 | 0.737 | 0.986 | 0.761 |
| 1 | 32 | 4.429 | 0.499 | 0.813 | 0.992 | 0.866 |
| 1 | 64 | 4.422 | 0.486 | 0.822 | 0.992 | 0.872 |
| 1 | 128 | 4.685 | 0.405 | 0.730 | 0.986 | 0.795 |
| 2 | 2 | 7.249 | 0.039 | 0.261 | 0.683 | 0.189 |
| 2 | 4 | 4.933 | 0.262 | 0.733 | 0.992 | 0.705 |
| 2 | 8 | 4.461 | 0.459 | 0.820 | 0.989 | 0.858 |
| 2 | 16 | 4.840 | 0.323 | 0.734 | 0.988 | 0.758 |
| 2 | 32 | 4.698 | 0.368 | 0.766 | 0.990 | 0.793 |
| 2 | 64 | 5.024 | 0.295 | 0.653 | 0.982 | 0.704 |
| 2 | 128 | 4.297 | 0.523 | 0.850 | 0.993 | 0.892 |
| 3 | 2 | 7.236 | 0.047 | 0.261 | 0.677 | 0.213 |
| 3 | 4 | 7.234 | 0.046 | 0.254 | 0.671 | 0.196 |
| 3 | 8 | 5.059 | 0.268 | 0.690 | 0.984 | 0.678 |
| 3 | 16 | 4.651 | 0.359 | 0.768 | 0.995 | 0.792 |
| 3 | 32 | 4.689 | 0.353 | 0.760 | 0.990 | 0.784 |
| 3 | 64 | 4.686 | 0.374 | 0.761 | 0.988 | 0.805 |
| 3 | 128 | 4.275 | 0.541 | 0.835 | 0.989 | 0.902 |
| 4 | 2 | 7.249 | 0.039 | 0.261 | 0.683 | 0.189 |
| 4 | 4 | 7.249 | 0.039 | 0.261 | 0.683 | 0.189 |
| 4 | 8 | 4.740 | 0.335 | 0.762 | 0.989 | 0.776 |
| 4 | 16 | 4.743 | 0.332 | 0.744 | 0.989 | 0.766 |
| 4 | 32 | 4.878 | 0.267 | 0.730 | 0.993 | 0.708 |
| 4 | 64 | 4.459 | 0.475 | 0.796 | 0.985 | 0.857 |
| 4 | 128 | 5.137 | 0.283 | 0.618 | 0.968 | 0.682 |
| 5 | 2 | 7.249 | 0.039 | 0.261 | 0.683 | 0.189 |
| 5 | 4 | 7.072 | 0.085 | 0.299 | 0.672 | 0.295 |
| 5 | 8 | 4.540 | 0.356 | 0.806 | 0.991 | 0.815 |
| 5 | 16 | 5.121 | 0.208 | 0.686 | 0.989 | 0.642 |
| 5 | 32 | 4.939 | 0.248 | 0.707 | 0.992 | 0.686 |
| 5 | 64 | 4.962 | 0.321 | 0.688 | 0.973 | 0.739 |
| 5 | 128 | 4.642 | 0.331 | 0.779 | 0.992 | 0.786 |

*Continued on next page*

*Table B.23 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 2 | 7.249 | 0.039 | 0.261 | 0.683 | 0.189 |
| 6 | 4 | 7.253 | 0.038 | 0.255 | 0.676 | 0.186 |
| 6 | 8 | 4.922 | 0.244 | 0.718 | 0.992 | 0.688 |
| 6 | 16 | 4.773 | 0.323 | 0.751 | 0.987 | 0.763 |
| 6 | 32 | 4.810 | 0.270 | 0.755 | 0.993 | 0.736 |
| 6 | 64 | 4.643 | 0.346 | 0.768 | 0.992 | 0.793 |
| 6 | 128 | 4.557 | 0.366 | 0.799 | 0.992 | 0.817 |

*Table B.24: Validation Metrics for N = 9 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.301 | 0.033 | 0.248 | 0.665 | 0.179 |
| 1 | 4 | 6.537 | 0.067 | 0.444 | 0.916 | 0.315 |
| 1 | 8 | 7.266 | 0.038 | 0.264 | 0.681 | 0.190 |
| 1 | 16 | 5.563 | 0.111 | 0.580 | 0.988 | 0.450 |
| 1 | 32 | 5.807 | 0.096 | 0.567 | 0.986 | 0.439 |
| 1 | 64 | 5.667 | 0.103 | 0.603 | 0.987 | 0.456 |
| 1 | 128 | 5.914 | 0.098 | 0.554 | 0.983 | 0.407 |
| 2 | 2 | 7.259 | 0.038 | 0.266 | 0.679 | 0.198 |
| 2 | 4 | 5.366 | 0.104 | 0.654 | 0.989 | 0.490 |
| 2 | 8 | 5.735 | 0.092 | 0.586 | 0.986 | 0.436 |
| 2 | 16 | 5.508 | 0.100 | 0.604 | 0.986 | 0.456 |
| 2 | 32 | 5.562 | 0.109 | 0.608 | 0.988 | 0.473 |
| 2 | 64 | 5.792 | 0.097 | 0.540 | 0.989 | 0.403 |
| 2 | 128 | 5.796 | 0.100 | 0.575 | 0.987 | 0.453 |
| 3 | 2 | 7.261 | 0.036 | 0.267 | 0.670 | 0.182 |
| 3 | 4 | 7.261 | 0.037 | 0.252 | 0.673 | 0.183 |
| 3 | 8 | 5.542 | 0.094 | 0.621 | 0.984 | 0.472 |
| 3 | 16 | 5.580 | 0.109 | 0.609 | 0.986 | 0.479 |
| 3 | 32 | 5.643 | 0.106 | 0.597 | 0.984 | 0.483 |
| 3 | 64 | 5.818 | 0.102 | 0.571 | 0.978 | 0.449 |
| 3 | 128 | 6.171 | 0.093 | 0.545 | 0.970 | 0.434 |
| 4 | 2 | 7.259 | 0.038 | 0.266 | 0.679 | 0.198 |
| 4 | 4 | 7.259 | 0.038 | 0.266 | 0.679 | 0.198 |
| 4 | 8 | 5.507 | 0.101 | 0.610 | 0.989 | 0.480 |

*Continued on next page*

*Table B.24 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 4 | 16 | 5.679 | 0.110 | 0.591 | 0.986 | 0.450 |
| 4 | 32 | 5.436 | 0.108 | 0.638 | 0.992 | 0.509 |
| 4 | 64 | 6.044 | 0.097 | 0.561 | 0.972 | 0.450 |
| 4 | 128 | 6.316 | 0.075 | 0.487 | 0.965 | 0.367 |
| 5 | 2 | 7.259 | 0.038 | 0.266 | 0.679 | 0.198 |
| 5 | 4 | 7.323 | 0.027 | 0.239 | 0.659 | 0.170 |
| 5 | 8 | 5.642 | 0.109 | 0.616 | 0.988 | 0.470 |
| 5 | 16 | 5.516 | 0.109 | 0.607 | 0.985 | 0.476 |
| 5 | 32 | 5.474 | 0.106 | 0.610 | 0.989 | 0.473 |
| 5 | 64 | 6.205 | 0.091 | 0.513 | 0.962 | 0.385 |
| 5 | 128 | 5.629 | 0.084 | 0.615 | 0.986 | 0.452 |
| 6 | 2 | 7.259 | 0.038 | 0.266 | 0.679 | 0.198 |
| 6 | 4 | 7.264 | 0.035 | 0.255 | 0.680 | 0.178 |
| 6 | 8 | 5.326 | 0.116 | 0.632 | 0.995 | 0.517 |
| 6 | 16 | 5.630 | 0.116 | 0.604 | 0.984 | 0.468 |
| 6 | 32 | 5.404 | 0.113 | 0.636 | 0.990 | 0.502 |
| 6 | 64 | 5.625 | 0.099 | 0.596 | 0.987 | 0.456 |
| 6 | 128 | 5.784 | 0.110 | 0.599 | 0.989 | 0.461 |

*Table B.25: Model Metrics for N = 10 (Distinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 51 | 14 | 46.2 | 32,319 |
| 1 | 4 | 51 | 108 | 280.2 | 57,939 |
| 1 | 8 | 51 | 40 | 121.7 | 109,203 |
| 1 | 16 | 51 | 33 | 47.7 | 211,827 |
| 1 | 32 | 51 | 30 | 66.7 | 417,459 |
| 1 | 64 | 51 | 23 | 98.6 | 830,259 |
| 1 | 128 | 51 | 22 | 175.9 | 1,662,003 |
| 2 | 2 | 51 | 11 | 42.5 | 32,325 |
| 2 | 4 | 51 | 74 | 220.7 | 57,959 |
| 2 | 8 | 51 | 48 | 181.5 | 109,275 |
| 2 | 16 | 51 | 36 | 67.4 | 212,099 |
| 2 | 32 | 51 | 30 | 82.1 | 418,515 |
| 2 | 64 | 51 | 20 | 107.1 | 834,419 |

*Continued on next page*

*Table B.25 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 2 | 128 | 51 | 18 | 183.0 | 1,678,515 |
| 3 | 2 | 51 | 20 | 82.9 | 32,331 |
| 3 | 4 | 51 | 13 | 57.0 | 57,979 |
| 3 | 8 | 51 | 37 | 170.7 | 109,347 |
| 3 | 16 | 51 | 33 | 175.7 | 212,371 |
| 3 | 32 | 51 | 29 | 148.6 | 419,571 |
| 3 | 64 | 51 | 20 | 155.8 | 838,579 |
| 3 | 128 | 51 | 18 | 248.4 | 1,695,027 |
| 4 | 2 | 51 | 11 | 59.1 | 32,337 |
| 4 | 4 | 51 | 15 | 75.5 | 57,999 |
| 4 | 8 | 51 | 12 | 78.5 | 109,419 |
| 4 | 16 | 51 | 27 | 170.7 | 212,643 |
| 4 | 32 | 51 | 27 | 167.4 | 420,627 |
| 4 | 64 | 51 | 25 | 222.8 | 842,739 |
| 4 | 128 | 51 | 20 | 321.7 | 1,711,539 |
| 5 | 2 | 51 | 44 | 227.1 | 32,343 |
| 5 | 4 | 51 | 28 | 166.0 | 58,019 |
| 5 | 8 | 51 | 54 | 311.8 | 109,491 |
| 5 | 16 | 51 | 33 | 200.4 | 212,915 |
| 5 | 32 | 51 | 24 | 159.7 | 421,683 |
| 5 | 64 | 51 | 26 | 251.6 | 846,899 |
| 5 | 128 | 51 | 22 | 458.8 | 1,728,051 |
| 6 | 2 | 51 | 43 | 245.1 | 32,349 |
| 6 | 4 | 51 | 19 | 120.5 | 58,039 |
| 6 | 8 | 51 | 30 | 202.4 | 109,563 |
| 6 | 16 | 51 | 32 | 246.6 | 213,187 |
| 6 | 32 | 51 | 28 | 209.5 | 422,739 |
| 6 | 64 | 51 | 27 | 305.7 | 851,059 |
| 6 | 128 | 51 | 25 | 568.8 | 1,744,563 |

*Table B.26: Training Metrics for N = 10 (Distinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.021 | 0.106 | 0.306 | 0.652 | 0.333 |
| 1 | 4 | 4.623 | 0.425 | 0.800 | 0.979 | 0.833 |

*Continued on next page*

*Table B.26 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 8 | 4.687 | 0.403 | 0.758 | 0.989 | 0.810 |
| 1 | 16 | 4.288 | 0.556 | 0.858 | 0.990 | 0.905 |
| 1 | 32 | 4.428 | 0.493 | 0.816 | 0.992 | 0.871 |
| 1 | 64 | 4.259 | 0.575 | 0.852 | 0.994 | 0.906 |
| 1 | 128 | 4.150 | 0.610 | 0.867 | 0.995 | 0.925 |
| 2 | 2 | 7.190 | 0.068 | 0.280 | 0.671 | 0.256 |
| 2 | 4 | 4.953 | 0.255 | 0.722 | 0.992 | 0.695 |
| 2 | 8 | 4.661 | 0.370 | 0.771 | 0.995 | 0.798 |
| 2 | 16 | 4.553 | 0.455 | 0.782 | 0.989 | 0.839 |
| 2 | 32 | 4.444 | 0.489 | 0.789 | 0.991 | 0.865 |
| 2 | 64 | 4.023 | 0.659 | 0.890 | 0.995 | 0.949 |
| 2 | 128 | 4.443 | 0.524 | 0.780 | 0.985 | 0.873 |
| 3 | 2 | 7.170 | 0.072 | 0.280 | 0.665 | 0.267 |
| 3 | 4 | 7.119 | 0.080 | 0.280 | 0.651 | 0.297 |
| 3 | 8 | 4.308 | 0.488 | 0.857 | 0.986 | 0.891 |
| 3 | 16 | 4.460 | 0.439 | 0.814 | 0.993 | 0.846 |
| 3 | 32 | 4.685 | 0.355 | 0.757 | 0.996 | 0.780 |
| 3 | 64 | 4.068 | 0.624 | 0.887 | 0.995 | 0.941 |
| 3 | 128 | 4.022 | 0.645 | 0.894 | 0.994 | 0.940 |
| 4 | 2 | 7.278 | 0.043 | 0.262 | 0.681 | 0.196 |
| 4 | 4 | 7.274 | 0.043 | 0.262 | 0.681 | 0.198 |
| 4 | 8 | 6.888 | 0.117 | 0.324 | 0.672 | 0.376 |
| 4 | 16 | 4.284 | 0.543 | 0.835 | 0.976 | 0.896 |
| 4 | 32 | 4.714 | 0.340 | 0.761 | 0.991 | 0.781 |
| 4 | 64 | 4.477 | 0.420 | 0.808 | 0.990 | 0.842 |
| 4 | 128 | 4.569 | 0.423 | 0.752 | 0.988 | 0.834 |
| 5 | 2 | 7.273 | 0.043 | 0.262 | 0.681 | 0.198 |
| 5 | 4 | 7.273 | 0.041 | 0.266 | 0.675 | 0.197 |
| 5 | 8 | 5.391 | 0.203 | 0.613 | 0.978 | 0.590 |
| 5 | 16 | 4.679 | 0.347 | 0.765 | 0.994 | 0.785 |
| 5 | 32 | 4.527 | 0.408 | 0.799 | 0.993 | 0.831 |
| 5 | 64 | 4.239 | 0.508 | 0.859 | 0.991 | 0.900 |
| 5 | 128 | 4.970 | 0.309 | 0.692 | 0.975 | 0.731 |
| 6 | 2 | 7.273 | 0.043 | 0.262 | 0.681 | 0.198 |
| 6 | 4 | 7.272 | 0.038 | 0.267 | 0.681 | 0.194 |
| 6 | 8 | 7.274 | 0.043 | 0.262 | 0.681 | 0.193 |

*Continued on next page*

*Table B.26 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 16 | 4.562 | 0.382 | 0.796 | 0.989 | 0.816 |
| 6 | 32 | 4.713 | 0.308 | 0.761 | 0.993 | 0.757 |
| 6 | 64 | 4.449 | 0.401 | 0.817 | 0.993 | 0.843 |
| 6 | 128 | 4.257 | 0.474 | 0.854 | 0.992 | 0.886 |

*Table B.27: Validation Metrics for N = 10 (Distinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.401 | 0.037 | 0.222 | 0.634 | 0.167 |
| 1 | 4 | 6.182 | 0.075 | 0.500 | 0.955 | 0.378 |
| 1 | 8 | 5.675 | 0.094 | 0.564 | 0.979 | 0.443 |
| 1 | 16 | 6.035 | 0.075 | 0.550 | 0.974 | 0.405 |
| 1 | 32 | 5.732 | 0.099 | 0.592 | 0.984 | 0.455 |
| 1 | 64 | 5.875 | 0.094 | 0.583 | 0.983 | 0.429 |
| 1 | 128 | 5.902 | 0.091 | 0.577 | 0.985 | 0.449 |
| 2 | 2 | 7.306 | 0.042 | 0.247 | 0.668 | 0.175 |
| 2 | 4 | 5.378 | 0.119 | 0.628 | 0.989 | 0.493 |
| 2 | 8 | 5.528 | 0.108 | 0.604 | 0.988 | 0.470 |
| 2 | 16 | 5.730 | 0.099 | 0.598 | 0.987 | 0.454 |
| 2 | 32 | 5.961 | 0.083 | 0.561 | 0.982 | 0.426 |
| 2 | 64 | 6.168 | 0.089 | 0.565 | 0.977 | 0.417 |
| 2 | 128 | 6.281 | 0.088 | 0.517 | 0.969 | 0.391 |
| 3 | 2 | 7.309 | 0.032 | 0.239 | 0.661 | 0.172 |
| 3 | 4 | 7.344 | 0.031 | 0.226 | 0.640 | 0.166 |
| 3 | 8 | 6.120 | 0.084 | 0.538 | 0.968 | 0.383 |
| 3 | 16 | 5.615 | 0.095 | 0.599 | 0.989 | 0.478 |
| 3 | 32 | 5.505 | 0.109 | 0.620 | 0.993 | 0.489 |
| 3 | 64 | 6.040 | 0.086 | 0.567 | 0.987 | 0.430 |
| 3 | 128 | 6.151 | 0.099 | 0.572 | 0.982 | 0.418 |
| 4 | 2 | 7.278 | 0.039 | 0.259 | 0.686 | 0.188 |
| 4 | 4 | 7.277 | 0.038 | 0.260 | 0.687 | 0.180 |
| 4 | 8 | 7.363 | 0.029 | 0.224 | 0.624 | 0.160 |
| 4 | 16 | 6.336 | 0.083 | 0.531 | 0.962 | 0.388 |
| 4 | 32 | 5.511 | 0.118 | 0.629 | 0.989 | 0.479 |
| 4 | 64 | 5.724 | 0.098 | 0.614 | 0.988 | 0.465 |

*Continued on next page*

*Table B.27 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 4 | 128 | 6.255 | 0.087 | 0.530 | 0.977 | 0.395 |
| 5 | 2 | 7.273 | 0.038 | 0.260 | 0.687 | 0.180 |
| 5 | 4 | 7.277 | 0.037 | 0.255 | 0.683 | 0.184 |
| 5 | 8 | 5.886 | 0.094 | 0.549 | 0.977 | 0.418 |
| 5 | 16 | 5.505 | 0.105 | 0.610 | 0.991 | 0.478 |
| 5 | 32 | 5.613 | 0.102 | 0.615 | 0.986 | 0.468 |
| 5 | 64 | 5.952 | 0.104 | 0.602 | 0.988 | 0.459 |
| 5 | 128 | 5.988 | 0.092 | 0.544 | 0.975 | 0.413 |
| 6 | 2 | 7.273 | 0.038 | 0.260 | 0.687 | 0.180 |
| 6 | 4 | 7.275 | 0.036 | 0.261 | 0.687 | 0.186 |
| 6 | 8 | 7.274 | 0.038 | 0.260 | 0.687 | 0.176 |
| 6 | 16 | 5.644 | 0.114 | 0.616 | 0.984 | 0.494 |
| 6 | 32 | 5.468 | 0.116 | 0.640 | 0.993 | 0.503 |
| 6 | 64 | 5.687 | 0.109 | 0.601 | 0.990 | 0.483 |
| 6 | 128 | 6.000 | 0.104 | 0.591 | 0.988 | 0.445 |

# Appendix C

# Network Results for Indistinguishable Permanents

*Table C.1: Model Metrics for N = 2 (Indistinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 72 | 126 | 48.9 | 10,452 |
| 1 | 4 | 72 | 199 | 73.9 | 11,496 |
| 1 | 8 | 72 | 236 | 87.7 | 13,608 |
| 1 | 16 | 72 | 157 | 213.9 | 17,928 |
| 1 | 32 | 72 | 147 | 244.3 | 26,952 |
| 1 | 64 | 72 | 120 | 284.6 | 46,536 |
| 1 | 128 | 72 | 104 | 281.7 | 91,848 |
| 2 | 2 | 72 | 27 | 13.1 | 10,458 |
| 2 | 4 | 72 | 231 | 91.2 | 11,516 |
| 2 | 8 | 72 | 199 | 80.2 | 13,680 |
| 2 | 16 | 72 | 126 | 184.2 | 18,200 |
| 2 | 32 | 72 | 83 | 144.4 | 28,008 |
| 2 | 64 | 72 | 83 | 208.6 | 50,696 |
| 2 | 128 | 72 | 58 | 185.7 | 108,360 |
| 3 | 2 | 72 | 38 | 20.4 | 10,464 |
| 3 | 4 | 72 | 270 | 124.2 | 11,536 |
| 3 | 8 | 72 | 176 | 81.3 | 13,752 |
| 3 | 16 | 72 | 113 | 194.0 | 18,472 |
| 3 | 32 | 72 | 71 | 144.1 | 29,064 |
| 3 | 64 | 72 | 51 | 159.0 | 54,856 |
| 3 | 128 | 72 | 50 | 204.1 | 124,872 |
| 4 | 2 | 72 | 90 | 48.2 | 10,470 |

*Continued on next page*

*Table C.1 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 4 | 4 | 72 | 157 | 79.9 | 11,556 |
| 4 | 8 | 72 | 124 | 64.0 | 13,824 |
| 4 | 16 | 72 | 102 | 194.4 | 18,744 |
| 4 | 32 | 72 | 76 | 176.3 | 30,120 |
| 4 | 64 | 72 | 74 | 263.0 | 59,016 |
| 4 | 128 | 72 | 56 | 287.8 | 141,384 |
| 5 | 2 | 72 | 74 | 42.9 | 10,476 |
| 5 | 4 | 72 | 86 | 49.3 | 11,576 |
| 5 | 8 | 72 | 115 | 61.9 | 13,896 |
| 5 | 16 | 72 | 101 | 219.8 | 19,016 |
| 5 | 32 | 72 | 59 | 157.4 | 31,176 |
| 5 | 64 | 72 | 58 | 241.6 | 63,176 |
| 5 | 128 | 72 | 45 | 273.1 | 157,896 |
| 6 | 2 | 72 | 69 | 46.8 | 10,482 |
| 6 | 4 | 72 | 21 | 16.5 | 11,596 |
| 6 | 8 | 72 | 98 | 57.4 | 13,968 |
| 6 | 16 | 72 | 83 | 206.0 | 19,288 |
| 6 | 32 | 72 | 64 | 185.9 | 32,232 |
| 6 | 64 | 72 | 58 | 284.8 | 67,336 |
| 6 | 128 | 72 | 41 | 301.8 | 174,408 |

*Table C.2: Training Metrics for N = 2 (Indistinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.366 | 0.059 | 0.265 | 0.641 | 0.237 |
| 1 | 4 | 6.594 | 0.100 | 0.406 | 0.815 | 0.362 |
| 1 | 8 | 6.184 | 0.143 | 0.496 | 0.817 | 0.464 |
| 1 | 16 | 6.032 | 0.166 | 0.511 | 0.822 | 0.500 |
| 1 | 32 | 6.034 | 0.162 | 0.509 | 0.805 | 0.493 |
| 1 | 64 | 6.049 | 0.149 | 0.516 | 0.811 | 0.479 |
| 1 | 128 | 5.955 | 0.157 | 0.529 | 0.831 | 0.502 |
| 2 | 2 | 7.718 | 0.038 | 0.215 | 0.576 | 0.172 |
| 2 | 4 | 6.282 | 0.107 | 0.453 | 0.843 | 0.392 |
| 2 | 8 | 6.123 | 0.131 | 0.532 | 0.824 | 0.468 |
| 2 | 16 | 6.011 | 0.136 | 0.530 | 0.820 | 0.463 |

*Continued on next page*

*Table C.2 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 2 | 32 | 5.995 | 0.136 | 0.524 | 0.829 | 0.473 |
| 2 | 64 | 6.022 | 0.152 | 0.532 | 0.802 | 0.492 |
| 2 | 128 | 5.933 | 0.143 | 0.527 | 0.833 | 0.473 |
| 3 | 2 | 7.707 | 0.040 | 0.219 | 0.577 | 0.176 |
| 3 | 4 | 7.079 | 0.080 | 0.315 | 0.714 | 0.286 |
| 3 | 8 | 6.026 | 0.116 | 0.513 | 0.835 | 0.428 |
| 3 | 16 | 6.003 | 0.126 | 0.514 | 0.820 | 0.450 |
| 3 | 32 | 5.931 | 0.136 | 0.540 | 0.818 | 0.481 |
| 3 | 64 | 6.064 | 0.114 | 0.493 | 0.834 | 0.421 |
| 3 | 128 | 5.922 | 0.148 | 0.535 | 0.821 | 0.492 |
| 4 | 2 | 7.850 | 0.031 | 0.202 | 0.548 | 0.147 |
| 4 | 4 | 5.988 | 0.139 | 0.530 | 0.821 | 0.464 |
| 4 | 8 | 6.033 | 0.136 | 0.530 | 0.817 | 0.478 |
| 4 | 16 | 5.945 | 0.143 | 0.536 | 0.815 | 0.475 |
| 4 | 32 | 5.935 | 0.145 | 0.543 | 0.818 | 0.492 |
| 4 | 64 | 5.866 | 0.149 | 0.549 | 0.822 | 0.495 |
| 4 | 128 | 5.884 | 0.160 | 0.540 | 0.816 | 0.507 |
| 5 | 2 | 7.850 | 0.031 | 0.202 | 0.548 | 0.147 |
| 5 | 4 | 7.850 | 0.031 | 0.202 | 0.548 | 0.147 |
| 5 | 8 | 5.891 | 0.159 | 0.555 | 0.816 | 0.505 |
| 5 | 16 | 5.921 | 0.163 | 0.541 | 0.807 | 0.510 |
| 5 | 32 | 5.947 | 0.143 | 0.539 | 0.818 | 0.492 |
| 5 | 64 | 5.933 | 0.150 | 0.524 | 0.810 | 0.492 |
| 5 | 128 | 5.914 | 0.145 | 0.536 | 0.833 | 0.480 |
| 6 | 2 | 7.850 | 0.031 | 0.202 | 0.548 | 0.147 |
| 6 | 4 | 7.850 | 0.031 | 0.202 | 0.548 | 0.147 |
| 6 | 8 | 6.235 | 0.129 | 0.498 | 0.801 | 0.444 |
| 6 | 16 | 5.856 | 0.153 | 0.549 | 0.824 | 0.495 |
| 6 | 32 | 5.861 | 0.154 | 0.554 | 0.818 | 0.507 |
| 6 | 64 | 5.933 | 0.140 | 0.520 | 0.820 | 0.474 |
| 6 | 128 | 6.016 | 0.120 | 0.521 | 0.834 | 0.438 |

*Table C.3: Validation Metrics for N = 2 (Indistinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.505 | 0.039 | 0.262 | 0.634 | 0.185 |
| 1 | 4 | 6.739 | 0.065 | 0.383 | 0.814 | 0.292 |
| 1 | 8 | 6.503 | 0.079 | 0.445 | 0.799 | 0.344 |
| 1 | 16 | 6.464 | 0.077 | 0.450 | 0.819 | 0.354 |
| 1 | 32 | 6.349 | 0.083 | 0.462 | 0.804 | 0.367 |
| 1 | 64 | 6.334 | 0.086 | 0.465 | 0.814 | 0.372 |
| 1 | 128 | 6.257 | 0.087 | 0.491 | 0.838 | 0.380 |
| 2 | 2 | 7.772 | 0.030 | 0.208 | 0.552 | 0.150 |
| 2 | 4 | 6.442 | 0.082 | 0.430 | 0.827 | 0.326 |
| 2 | 8 | 6.326 | 0.090 | 0.508 | 0.814 | 0.394 |
| 2 | 16 | 6.151 | 0.088 | 0.506 | 0.843 | 0.381 |
| 2 | 32 | 6.189 | 0.088 | 0.500 | 0.850 | 0.393 |
| 2 | 64 | 6.217 | 0.108 | 0.516 | 0.823 | 0.426 |
| 2 | 128 | 6.152 | 0.084 | 0.499 | 0.858 | 0.385 |
| 3 | 2 | 7.777 | 0.025 | 0.216 | 0.569 | 0.133 |
| 3 | 4 | 7.258 | 0.047 | 0.290 | 0.703 | 0.219 |
| 3 | 8 | 6.230 | 0.080 | 0.477 | 0.819 | 0.346 |
| 3 | 16 | 6.134 | 0.077 | 0.504 | 0.832 | 0.370 |
| 3 | 32 | 6.069 | 0.100 | 0.528 | 0.846 | 0.410 |
| 3 | 64 | 6.199 | 0.078 | 0.488 | 0.857 | 0.364 |
| 3 | 128 | 6.168 | 0.094 | 0.502 | 0.841 | 0.387 |
| 4 | 2 | 7.884 | 0.027 | 0.191 | 0.531 | 0.144 |
| 4 | 4 | 6.199 | 0.090 | 0.503 | 0.803 | 0.407 |
| 4 | 8 | 6.223 | 0.098 | 0.502 | 0.812 | 0.406 |
| 4 | 16 | 6.059 | 0.099 | 0.525 | 0.839 | 0.418 |
| 4 | 32 | 6.089 | 0.098 | 0.533 | 0.844 | 0.430 |
| 4 | 64 | 6.030 | 0.098 | 0.538 | 0.845 | 0.433 |
| 4 | 128 | 6.107 | 0.100 | 0.515 | 0.836 | 0.428 |
| 5 | 2 | 7.884 | 0.027 | 0.191 | 0.531 | 0.144 |
| 5 | 4 | 7.884 | 0.027 | 0.191 | 0.531 | 0.144 |
| 5 | 8 | 6.163 | 0.089 | 0.519 | 0.799 | 0.427 |
| 5 | 16 | 6.084 | 0.104 | 0.538 | 0.826 | 0.451 |
| 5 | 32 | 6.087 | 0.105 | 0.526 | 0.842 | 0.433 |
| 5 | 64 | 6.135 | 0.108 | 0.517 | 0.834 | 0.428 |
| 5 | 128 | 6.130 | 0.082 | 0.499 | 0.846 | 0.388 |

*Continued on next page*

*Table C.3 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 2 | 7.884 | 0.027 | 0.191 | 0.531 | 0.144 |
| 6 | 4 | 7.882 | 0.027 | 0.191 | 0.531 | 0.144 |
| 6 | 8 | 6.389 | 0.099 | 0.481 | 0.791 | 0.408 |
| 6 | 16 | 6.032 | 0.099 | 0.533 | 0.848 | 0.435 |
| 6 | 32 | 6.063 | 0.092 | 0.534 | 0.840 | 0.420 |
| 6 | 64 | 6.115 | 0.103 | 0.512 | 0.847 | 0.401 |
| 6 | 128 | 6.204 | 0.071 | 0.502 | 0.852 | 0.359 |

*Table C.4: Model Metrics for N = 3 (Indistinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 78 | 77 | 55.4 | 12,506 |
| 1 | 4 | 78 | 123 | 90.4 | 14,830 |
| 1 | 8 | 78 | 35 | 34.8 | 19,502 |
| 1 | 16 | 78 | 28 | 123.7 | 28,942 |
| 1 | 32 | 78 | 50 | 232.7 | 48,206 |
| 1 | 64 | 78 | 41 | 223.6 | 88,270 |
| 1 | 128 | 78 | 31 | 166.5 | 174,542 |
| 2 | 2 | 78 | 15 | 17.7 | 12,512 |
| 2 | 4 | 78 | 158 | 123.1 | 14,850 |
| 2 | 8 | 78 | 80 | 79.8 | 19,574 |
| 2 | 16 | 78 | 58 | 211.5 | 29,214 |
| 2 | 32 | 78 | 37 | 171.8 | 49,262 |
| 2 | 64 | 78 | 36 | 204.5 | 92,430 |
| 2 | 128 | 78 | 33 | 221.4 | 191,054 |
| 3 | 2 | 78 | 26 | 30.3 | 12,518 |
| 3 | 4 | 78 | 137 | 120.6 | 14,870 |
| 3 | 8 | 78 | 67 | 79.4 | 19,646 |
| 3 | 16 | 78 | 47 | 246.6 | 29,486 |
| 3 | 32 | 78 | 43 | 272.1 | 50,318 |
| 3 | 64 | 78 | 34 | 240.7 | 96,590 |
| 3 | 128 | 78 | 29 | 266.6 | 207,566 |
| 4 | 2 | 78 | 93 | 89.7 | 12,524 |
| 4 | 4 | 78 | 83 | 85.4 | 14,890 |
| 4 | 8 | 78 | 71 | 89.4 | 19,718 |

*Continued on next page*

*Table C.4 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 4 | 16 | 78 | 47 | 275.0 | 29,758 |
| 4 | 32 | 78 | 38 | 270.7 | 51,374 |
| 4 | 64 | 78 | 30 | 252.1 | 100,750 |
| 4 | 128 | 78 | 33 | 352.1 | 224,078 |
| 5 | 2 | 78 | 91 | 99.8 | 12,530 |
| 5 | 4 | 78 | 17 | 27.5 | 14,910 |
| 5 | 8 | 78 | 68 | 99.0 | 19,790 |
| 5 | 16 | 78 | 42 | 288.6 | 30,030 |
| 5 | 32 | 78 | 42 | 335.6 | 52,430 |
| 5 | 64 | 78 | 39 | 397.9 | 104,910 |
| 5 | 128 | 78 | 34 | 413.9 | 240,590 |
| 6 | 2 | 78 | 91 | 108.7 | 12,536 |
| 6 | 4 | 78 | 24 | 36.9 | 14,930 |
| 6 | 8 | 78 | 70 | 110.6 | 19,862 |
| 6 | 16 | 78 | 39 | 298.1 | 30,302 |
| 6 | 32 | 78 | 36 | 323.9 | 53,486 |
| 6 | 64 | 78 | 34 | 374.4 | 109,070 |
| 6 | 128 | 78 | 38 | 544.2 | 257,102 |

*Table C.5: Training Metrics for N = 3 (Indistinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.571 | 0.059 | 0.249 | 0.596 | 0.234 |
| 1 | 4 | 7.092 | 0.107 | 0.322 | 0.671 | 0.337 |
| 1 | 8 | 7.381 | 0.061 | 0.266 | 0.647 | 0.240 |
| 1 | 16 | 7.380 | 0.071 | 0.279 | 0.644 | 0.264 |
| 1 | 32 | 7.060 | 0.100 | 0.323 | 0.687 | 0.335 |
| 1 | 64 | 7.154 | 0.100 | 0.313 | 0.667 | 0.316 |
| 1 | 128 | 7.189 | 0.091 | 0.309 | 0.657 | 0.314 |
| 2 | 2 | 8.015 | 0.026 | 0.180 | 0.489 | 0.141 |
| 2 | 4 | 7.143 | 0.089 | 0.315 | 0.689 | 0.307 |
| 2 | 8 | 7.059 | 0.101 | 0.336 | 0.684 | 0.340 |
| 2 | 16 | 7.079 | 0.098 | 0.323 | 0.689 | 0.332 |
| 2 | 32 | 7.042 | 0.107 | 0.337 | 0.690 | 0.344 |
| 2 | 64 | 6.955 | 0.116 | 0.352 | 0.695 | 0.366 |

*Continued on next page*

*Table C.5 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 2 | 128 | 6.960 | 0.118 | 0.355 | 0.693 | 0.362 |
| 3 | 2 | 8.015 | 0.029 | 0.185 | 0.513 | 0.138 |
| 3 | 4 | 7.117 | 0.086 | 0.322 | 0.699 | 0.310 |
| 3 | 8 | 7.087 | 0.090 | 0.326 | 0.703 | 0.320 |
| 3 | 16 | 7.078 | 0.100 | 0.325 | 0.673 | 0.338 |
| 3 | 32 | 6.994 | 0.104 | 0.341 | 0.704 | 0.349 |
| 3 | 64 | 7.024 | 0.104 | 0.337 | 0.689 | 0.337 |
| 3 | 128 | 6.985 | 0.106 | 0.345 | 0.700 | 0.349 |
| 4 | 2 | 8.017 | 0.029 | 0.186 | 0.513 | 0.137 |
| 4 | 4 | 7.183 | 0.086 | 0.317 | 0.698 | 0.310 |
| 4 | 8 | 7.032 | 0.099 | 0.328 | 0.705 | 0.330 |
| 4 | 16 | 7.042 | 0.098 | 0.333 | 0.705 | 0.324 |
| 4 | 32 | 7.024 | 0.101 | 0.327 | 0.696 | 0.326 |
| 4 | 64 | 6.996 | 0.100 | 0.338 | 0.709 | 0.336 |
| 4 | 128 | 6.956 | 0.106 | 0.348 | 0.707 | 0.346 |
| 5 | 2 | 8.017 | 0.029 | 0.186 | 0.513 | 0.137 |
| 5 | 4 | 8.019 | 0.026 | 0.182 | 0.511 | 0.131 |
| 5 | 8 | 7.073 | 0.092 | 0.327 | 0.695 | 0.321 |
| 5 | 16 | 7.070 | 0.092 | 0.318 | 0.698 | 0.312 |
| 5 | 32 | 6.979 | 0.103 | 0.339 | 0.706 | 0.343 |
| 5 | 64 | 6.860 | 0.121 | 0.356 | 0.699 | 0.374 |
| 5 | 128 | 7.019 | 0.095 | 0.333 | 0.714 | 0.320 |
| 6 | 2 | 8.017 | 0.029 | 0.186 | 0.513 | 0.137 |
| 6 | 4 | 8.019 | 0.030 | 0.183 | 0.516 | 0.138 |
| 6 | 8 | 7.025 | 0.106 | 0.343 | 0.695 | 0.341 |
| 6 | 16 | 7.081 | 0.086 | 0.326 | 0.696 | 0.307 |
| 6 | 32 | 7.028 | 0.091 | 0.326 | 0.707 | 0.318 |
| 6 | 64 | 7.003 | 0.095 | 0.340 | 0.708 | 0.334 |
| 6 | 128 | 6.963 | 0.102 | 0.344 | 0.711 | 0.347 |

*Table C.6: Validation Metrics for N = 3 (Indistinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.709 | 0.033 | 0.212 | 0.584 | 0.162 |
| 1 | 4 | 7.515 | 0.040 | 0.242 | 0.641 | 0.186 |

*Continued on next page*

*Table C.6 – continued from previous page*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 8 | 7.487 | 0.044 | 0.234 | 0.643 | 0.190 |
| 1 | 16 | 7.525 | 0.041 | 0.261 | 0.640 | 0.200 |
| 1 | 32 | 7.489 | 0.043 | 0.276 | 0.652 | 0.191 |
| 1 | 64 | 7.495 | 0.038 | 0.257 | 0.636 | 0.191 |
| 1 | 128 | 7.525 | 0.045 | 0.260 | 0.645 | 0.189 |
| 2 | 2 | 8.012 | 0.020 | 0.169 | 0.468 | 0.123 |
| 2 | 4 | 7.386 | 0.044 | 0.266 | 0.683 | 0.197 |
| 2 | 8 | 7.455 | 0.039 | 0.263 | 0.657 | 0.197 |
| 2 | 16 | 7.423 | 0.044 | 0.274 | 0.673 | 0.199 |
| 2 | 32 | 7.493 | 0.040 | 0.266 | 0.655 | 0.184 |
| 2 | 64 | 7.516 | 0.047 | 0.263 | 0.661 | 0.199 |
| 2 | 128 | 7.462 | 0.040 | 0.271 | 0.668 | 0.192 |
| 3 | 2 | 8.015 | 0.024 | 0.191 | 0.513 | 0.124 |
| 3 | 4 | 7.333 | 0.049 | 0.279 | 0.693 | 0.208 |
| 3 | 8 | 7.347 | 0.041 | 0.289 | 0.677 | 0.203 |
| 3 | 16 | 7.459 | 0.043 | 0.276 | 0.656 | 0.191 |
| 3 | 32 | 7.428 | 0.043 | 0.277 | 0.688 | 0.198 |
| 3 | 64 | 7.426 | 0.044 | 0.262 | 0.662 | 0.203 |
| 3 | 128 | 7.488 | 0.038 | 0.255 | 0.656 | 0.187 |
| 4 | 2 | 8.013 | 0.024 | 0.190 | 0.513 | 0.123 |
| 4 | 4 | 7.382 | 0.048 | 0.272 | 0.682 | 0.212 |
| 4 | 8 | 7.325 | 0.045 | 0.280 | 0.676 | 0.209 |
| 4 | 16 | 7.338 | 0.038 | 0.273 | 0.688 | 0.204 |
| 4 | 32 | 7.404 | 0.049 | 0.254 | 0.671 | 0.197 |
| 4 | 64 | 7.399 | 0.048 | 0.274 | 0.680 | 0.200 |
| 4 | 128 | 7.456 | 0.043 | 0.282 | 0.670 | 0.210 |
| 5 | 2 | 8.013 | 0.024 | 0.190 | 0.513 | 0.123 |
| 5 | 4 | 8.015 | 0.031 | 0.192 | 0.519 | 0.132 |
| 5 | 8 | 7.350 | 0.049 | 0.280 | 0.685 | 0.202 |
| 5 | 16 | 7.386 | 0.052 | 0.262 | 0.677 | 0.209 |
| 5 | 32 | 7.409 | 0.041 | 0.273 | 0.677 | 0.205 |
| 5 | 64 | 7.443 | 0.048 | 0.284 | 0.684 | 0.205 |
| 5 | 128 | 7.360 | 0.046 | 0.289 | 0.691 | 0.198 |
| 6 | 2 | 8.013 | 0.024 | 0.190 | 0.513 | 0.123 |
| 6 | 4 | 8.021 | 0.020 | 0.179 | 0.514 | 0.129 |
| 6 | 8 | 7.420 | 0.048 | 0.276 | 0.662 | 0.208 |

*Continued on next page*

*Table C.6 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 16 | 7.344 | 0.046 | 0.274 | 0.691 | 0.209 |
| 6 | 32 | 7.413 | 0.043 | 0.265 | 0.698 | 0.202 |
| 6 | 64 | 7.405 | 0.048 | 0.267 | 0.684 | 0.208 |
| 6 | 128 | 7.424 | 0.043 | 0.263 | 0.680 | 0.193 |

*Table C.7: Model Metrics for N = 4 (Indistinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 82 | 42 | 77.5 | 14,814 |
| 1 | 4 | 82 | 72 | 153.3 | 18,930 |
| 1 | 8 | 82 | 36 | 90.1 | 27,186 |
| 1 | 16 | 82 | 24 | 14.9 | 43,794 |
| 1 | 32 | 82 | 20 | 15.6 | 77,394 |
| 1 | 64 | 82 | 20 | 20.3 | 146,130 |
| 1 | 128 | 82 | 18 | 40.6 | 289,746 |
| 2 | 2 | 82 | 38 | 70.8 | 14,820 |
| 2 | 4 | 82 | 51 | 116.2 | 18,950 |
| 2 | 8 | 82 | 27 | 75.1 | 27,258 |
| 2 | 16 | 82 | 26 | 18.6 | 44,066 |
| 2 | 32 | 82 | 20 | 19.2 | 78,450 |
| 2 | 64 | 82 | 18 | 21.9 | 150,290 |
| 2 | 128 | 82 | 17 | 45.8 | 306,258 |
| 3 | 2 | 82 | 80 | 159.7 | 14,826 |
| 3 | 4 | 82 | 46 | 134.3 | 18,970 |
| 3 | 8 | 82 | 37 | 115.2 | 27,330 |
| 3 | 16 | 82 | 33 | 24.1 | 44,338 |
| 3 | 32 | 82 | 21 | 19.2 | 79,506 |
| 3 | 64 | 82 | 18 | 23.8 | 154,450 |
| 3 | 128 | 82 | 17 | 58.1 | 322,770 |
| 4 | 2 | 82 | 78 | 168.7 | 14,832 |
| 4 | 4 | 82 | 34 | 102.6 | 18,990 |
| 4 | 8 | 82 | 33 | 112.4 | 27,402 |
| 4 | 16 | 82 | 35 | 27.3 | 44,610 |
| 4 | 32 | 82 | 28 | 32.1 | 80,562 |
| 4 | 64 | 82 | 21 | 37.7 | 158,610 |

*Continued on next page*

*Table C.7 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 4 | 128 | 82 | 23 | 83.8 | 339,282 |
| 5 | 2 | 82 | 73 | 169.6 | 14,838 |
| 5 | 4 | 82 | 19 | 69.3 | 19,010 |
| 5 | 8 | 82 | 40 | 145.8 | 27,474 |
| 5 | 16 | 82 | 33 | 120.7 | 44,882 |
| 5 | 32 | 82 | 22 | 102.3 | 81,618 |
| 5 | 64 | 82 | 22 | 107.7 | 162,770 |
| 5 | 128 | 82 | 28 | 75.7 | 355,794 |
| 6 | 2 | 82 | 19 | 58.8 | 14,844 |
| 6 | 4 | 82 | 23 | 90.9 | 19,030 |
| 6 | 8 | 82 | 32 | 126.6 | 27,546 |
| 6 | 16 | 82 | 42 | 165.0 | 45,154 |
| 6 | 32 | 82 | 27 | 140.6 | 82,674 |
| 6 | 64 | 82 | 23 | 136.9 | 166,930 |
| 6 | 128 | 82 | 20 | 62.7 | 372,306 |

*Table C.8: Training Metrics for N = 4 (Indistinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 7.936 | 0.046 | 0.201 | 0.507 | 0.185 |
| 1 | 4 | 7.357 | 0.102 | 0.295 | 0.598 | 0.319 |
| 1 | 8 | 7.526 | 0.063 | 0.258 | 0.613 | 0.232 |
| 1 | 16 | 7.427 | 0.082 | 0.268 | 0.604 | 0.279 |
| 1 | 32 | 7.432 | 0.082 | 0.262 | 0.596 | 0.274 |
| 1 | 64 | 7.376 | 0.084 | 0.275 | 0.607 | 0.304 |
| 1 | 128 | 7.405 | 0.088 | 0.274 | 0.596 | 0.291 |
| 2 | 2 | 7.789 | 0.055 | 0.221 | 0.533 | 0.217 |
| 2 | 4 | 7.548 | 0.057 | 0.245 | 0.608 | 0.222 |
| 2 | 8 | 7.481 | 0.071 | 0.266 | 0.615 | 0.252 |
| 2 | 16 | 7.412 | 0.088 | 0.270 | 0.601 | 0.293 |
| 2 | 32 | 7.279 | 0.114 | 0.306 | 0.603 | 0.356 |
| 2 | 64 | 7.383 | 0.088 | 0.281 | 0.603 | 0.301 |
| 2 | 128 | 7.276 | 0.101 | 0.294 | 0.619 | 0.340 |
| 3 | 2 | 8.099 | 0.026 | 0.171 | 0.489 | 0.128 |
| 3 | 4 | 7.494 | 0.080 | 0.263 | 0.583 | 0.274 |

*Continued on next page*

*Table C.8 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 3 | 8 | 7.329 | 0.097 | 0.297 | 0.620 | 0.317 |
| 3 | 16 | 7.332 | 0.089 | 0.288 | 0.618 | 0.299 |
| 3 | 32 | 7.415 | 0.080 | 0.268 | 0.608 | 0.284 |
| 3 | 64 | 7.418 | 0.080 | 0.276 | 0.601 | 0.279 |
| 3 | 128 | 7.290 | 0.097 | 0.297 | 0.614 | 0.321 |
| 4 | 2 | 8.099 | 0.026 | 0.171 | 0.489 | 0.128 |
| 4 | 4 | 7.894 | 0.050 | 0.201 | 0.509 | 0.197 |
| 4 | 8 | 7.511 | 0.063 | 0.261 | 0.618 | 0.237 |
| 4 | 16 | 7.387 | 0.076 | 0.269 | 0.618 | 0.271 |
| 4 | 32 | 7.409 | 0.071 | 0.266 | 0.606 | 0.264 |
| 4 | 64 | 7.308 | 0.096 | 0.295 | 0.613 | 0.302 |
| 4 | 128 | 7.240 | 0.105 | 0.306 | 0.625 | 0.331 |
| 5 | 2 | 8.099 | 0.027 | 0.172 | 0.477 | 0.128 |
| 5 | 4 | 8.094 | 0.025 | 0.167 | 0.463 | 0.127 |
| 5 | 8 | 7.421 | 0.076 | 0.273 | 0.620 | 0.264 |
| 5 | 16 | 7.394 | 0.078 | 0.273 | 0.627 | 0.269 |
| 5 | 32 | 7.424 | 0.077 | 0.272 | 0.611 | 0.273 |
| 5 | 64 | 7.241 | 0.106 | 0.300 | 0.626 | 0.328 |
| 5 | 128 | 7.367 | 0.083 | 0.279 | 0.614 | 0.284 |
| 6 | 2 | 8.100 | 0.026 | 0.171 | 0.489 | 0.127 |
| 6 | 4 | 8.098 | 0.027 | 0.170 | 0.477 | 0.127 |
| 6 | 8 | 7.346 | 0.080 | 0.280 | 0.615 | 0.285 |
| 6 | 16 | 7.245 | 0.096 | 0.297 | 0.636 | 0.313 |
| 6 | 32 | 7.284 | 0.093 | 0.301 | 0.624 | 0.309 |
| 6 | 64 | 7.177 | 0.109 | 0.315 | 0.629 | 0.359 |
| 6 | 128 | 7.378 | 0.077 | 0.267 | 0.616 | 0.277 |

*Table C.9: Validation Metrics for N = 4 (Indistinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.068 | 0.030 | 0.176 | 0.509 | 0.127 |
| 1 | 4 | 7.857 | 0.030 | 0.200 | 0.568 | 0.155 |
| 1 | 8 | 7.646 | 0.040 | 0.225 | 0.599 | 0.172 |
| 1 | 16 | 7.720 | 0.038 | 0.248 | 0.590 | 0.164 |
| 1 | 32 | 7.718 | 0.040 | 0.229 | 0.587 | 0.175 |

*Continued on next page*

*Table C.9 – continued from previous page*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 64 | 7.739 | 0.037 | 0.232 | 0.596 | 0.167 |
| 1 | 128 | 7.806 | 0.032 | 0.204 | 0.581 | 0.160 |
| 2 | 2 | 7.975 | 0.029 | 0.189 | 0.515 | 0.140 |
| 2 | 4 | 7.661 | 0.044 | 0.231 | 0.594 | 0.169 |
| 2 | 8 | 7.653 | 0.040 | 0.239 | 0.596 | 0.176 |
| 2 | 16 | 7.744 | 0.034 | 0.220 | 0.584 | 0.167 |
| 2 | 32 | 7.803 | 0.039 | 0.235 | 0.579 | 0.164 |
| 2 | 64 | 7.772 | 0.039 | 0.246 | 0.588 | 0.165 |
| 2 | 128 | 7.770 | 0.041 | 0.224 | 0.576 | 0.177 |
| 3 | 2 | 8.104 | 0.026 | 0.168 | 0.497 | 0.126 |
| 3 | 4 | 7.800 | 0.031 | 0.211 | 0.542 | 0.153 |
| 3 | 8 | 7.774 | 0.032 | 0.216 | 0.577 | 0.157 |
| 3 | 16 | 7.669 | 0.040 | 0.238 | 0.603 | 0.181 |
| 3 | 32 | 7.712 | 0.039 | 0.238 | 0.610 | 0.164 |
| 3 | 64 | 7.725 | 0.032 | 0.242 | 0.594 | 0.168 |
| 3 | 128 | 7.759 | 0.036 | 0.218 | 0.580 | 0.163 |
| 4 | 2 | 8.104 | 0.026 | 0.168 | 0.497 | 0.126 |
| 4 | 4 | 8.114 | 0.021 | 0.167 | 0.470 | 0.108 |
| 4 | 8 | 7.645 | 0.038 | 0.223 | 0.589 | 0.171 |
| 4 | 16 | 7.657 | 0.042 | 0.249 | 0.616 | 0.170 |
| 4 | 32 | 7.661 | 0.037 | 0.239 | 0.596 | 0.173 |
| 4 | 64 | 7.731 | 0.033 | 0.239 | 0.591 | 0.170 |
| 4 | 128 | 7.763 | 0.033 | 0.208 | 0.566 | 0.161 |
| 5 | 2 | 8.104 | 0.029 | 0.177 | 0.488 | 0.126 |
| 5 | 4 | 8.105 | 0.025 | 0.178 | 0.469 | 0.127 |
| 5 | 8 | 7.622 | 0.042 | 0.236 | 0.590 | 0.189 |
| 5 | 16 | 7.678 | 0.031 | 0.225 | 0.599 | 0.173 |
| 5 | 32 | 7.700 | 0.032 | 0.230 | 0.592 | 0.164 |
| 5 | 64 | 7.784 | 0.037 | 0.214 | 0.577 | 0.162 |
| 5 | 128 | 7.677 | 0.036 | 0.231 | 0.591 | 0.159 |
| 6 | 2 | 8.105 | 0.026 | 0.168 | 0.497 | 0.124 |
| 6 | 4 | 8.104 | 0.028 | 0.176 | 0.489 | 0.132 |
| 6 | 8 | 7.680 | 0.038 | 0.230 | 0.588 | 0.170 |
| 6 | 16 | 7.644 | 0.032 | 0.231 | 0.606 | 0.170 |
| 6 | 32 | 7.683 | 0.035 | 0.217 | 0.585 | 0.176 |
| 6 | 64 | 7.791 | 0.033 | 0.208 | 0.568 | 0.169 |

*Continued on next page*

*Table C.9 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 128 | 7.645 | 0.038 | 0.227 | 0.604 | 0.161 |

*Table C.10: Model Metrics for N = 5 (Indistinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 83 | 19 | 69.8 | 17,247 |
| 1 | 4 | 83 | 14 | 62.9 | 23,667 |
| 1 | 8 | 83 | 32 | 157.8 | 36,531 |
| 1 | 16 | 83 | 25 | 25.3 | 62,355 |
| 1 | 32 | 83 | 22 | 26.0 | 114,387 |
| 1 | 64 | 83 | 20 | 32.0 | 219,987 |
| 1 | 128 | 83 | 17 | 47.5 | 437,331 |
| 2 | 2 | 83 | 50 | 167.7 | 17,253 |
| 2 | 4 | 83 | 30 | 133.9 | 23,687 |
| 2 | 8 | 83 | 27 | 149.6 | 36,603 |
| 2 | 16 | 83 | 25 | 32.3 | 62,627 |
| 2 | 32 | 83 | 21 | 31.6 | 115,443 |
| 2 | 64 | 83 | 18 | 36.8 | 224,147 |
| 2 | 128 | 83 | 17 | 71.3 | 453,843 |
| 3 | 2 | 83 | 21 | 83.7 | 17,259 |
| 3 | 4 | 83 | 97 | 414.2 | 23,707 |
| 3 | 8 | 83 | 31 | 193.7 | 36,675 |
| 3 | 16 | 83 | 24 | 36.5 | 62,899 |
| 3 | 32 | 83 | 20 | 49.1 | 116,499 |
| 3 | 64 | 83 | 18 | 49.5 | 228,307 |
| 3 | 128 | 83 | 19 | 72.5 | 470,355 |
| 4 | 2 | 83 | 16 | 75.6 | 17,265 |
| 4 | 4 | 83 | 16 | 97.9 | 23,727 |
| 4 | 8 | 83 | 38 | 244.5 | 36,747 |
| 4 | 16 | 83 | 27 | 52.1 | 63,171 |
| 4 | 32 | 83 | 21 | 48.8 | 117,555 |
| 4 | 64 | 83 | 19 | 57.2 | 232,467 |
| 4 | 128 | 83 | 18 | 80.5 | 486,867 |
| 5 | 2 | 83 | 12 | 72.4 | 17,271 |
| 5 | 4 | 83 | 42 | 225.0 | 23,747 |

*Continued on next page*

*Table C.10 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 5 | 8 | 83 | 22 | 166.1 | 36,819 |
| 5 | 16 | 83 | 25 | 130.1 | 63,443 |
| 5 | 32 | 83 | 26 | 160.7 | 118,611 |
| 5 | 64 | 83 | 18 | 119.9 | 236,627 |
| 5 | 128 | 83 | 18 | 77.7 | 503,379 |
| 6 | 2 | 83 | 104 | 419.4 | 17,277 |
| 6 | 4 | 83 | 18 | 113.4 | 23,767 |
| 6 | 8 | 83 | 40 | 335.0 | 36,891 |
| 6 | 16 | 83 | 33 | 191.9 | 63,715 |
| 6 | 32 | 83 | 20 | 140.8 | 119,667 |
| 6 | 64 | 83 | 20 | 146.2 | 240,787 |
| 6 | 128 | 83 | 20 | 101.2 | 519,891 |

*Table C.11: Training Metrics for N = 5 (Indistinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.146 | 0.030 | 0.168 | 0.475 | 0.133 |
| 1 | 4 | 8.006 | 0.057 | 0.205 | 0.480 | 0.215 |
| 1 | 8 | 7.443 | 0.109 | 0.287 | 0.566 | 0.323 |
| 1 | 16 | 7.418 | 0.098 | 0.284 | 0.571 | 0.321 |
| 1 | 32 | 7.316 | 0.120 | 0.302 | 0.573 | 0.369 |
| 1 | 64 | 7.209 | 0.135 | 0.320 | 0.584 | 0.396 |
| 1 | 128 | 7.204 | 0.138 | 0.320 | 0.596 | 0.392 |
| 2 | 2 | 7.814 | 0.068 | 0.223 | 0.521 | 0.233 |
| 2 | 4 | 7.880 | 0.063 | 0.215 | 0.496 | 0.229 |
| 2 | 8 | 7.644 | 0.061 | 0.233 | 0.571 | 0.227 |
| 2 | 16 | 7.386 | 0.110 | 0.289 | 0.570 | 0.339 |
| 2 | 32 | 7.225 | 0.134 | 0.325 | 0.587 | 0.392 |
| 2 | 64 | 7.281 | 0.130 | 0.319 | 0.582 | 0.376 |
| 2 | 128 | 7.225 | 0.138 | 0.330 | 0.587 | 0.380 |
| 3 | 2 | 8.156 | 0.025 | 0.167 | 0.479 | 0.126 |
| 3 | 4 | 8.155 | 0.025 | 0.167 | 0.479 | 0.126 |
| 3 | 8 | 7.276 | 0.132 | 0.316 | 0.575 | 0.379 |
| 3 | 16 | 7.392 | 0.104 | 0.283 | 0.581 | 0.331 |
| 3 | 32 | 7.163 | 0.139 | 0.334 | 0.588 | 0.407 |

*Continued on next page*

*Table C.11 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 3 | 64 | 7.206 | 0.139 | 0.321 | 0.583 | 0.397 |
| 3 | 128 | 6.830 | 0.194 | 0.399 | 0.623 | 0.496 |
| 4 | 2 | 8.136 | 0.038 | 0.179 | 0.476 | 0.148 |
| 4 | 4 | 8.140 | 0.033 | 0.173 | 0.467 | 0.143 |
| 4 | 8 | 7.604 | 0.073 | 0.246 | 0.571 | 0.255 |
| 4 | 16 | 7.523 | 0.078 | 0.260 | 0.560 | 0.284 |
| 4 | 32 | 7.352 | 0.110 | 0.292 | 0.577 | 0.344 |
| 4 | 64 | 7.384 | 0.108 | 0.282 | 0.581 | 0.330 |
| 4 | 128 | 7.032 | 0.156 | 0.362 | 0.607 | 0.438 |
| 5 | 2 | 8.118 | 0.034 | 0.176 | 0.471 | 0.157 |
| 5 | 4 | 7.781 | 0.068 | 0.233 | 0.531 | 0.237 |
| 5 | 8 | 7.629 | 0.069 | 0.240 | 0.555 | 0.243 |
| 5 | 16 | 7.717 | 0.064 | 0.230 | 0.536 | 0.234 |
| 5 | 32 | 7.495 | 0.083 | 0.263 | 0.572 | 0.287 |
| 5 | 64 | 7.441 | 0.096 | 0.274 | 0.575 | 0.312 |
| 5 | 128 | 7.409 | 0.102 | 0.287 | 0.568 | 0.323 |
| 6 | 2 | 8.155 | 0.025 | 0.167 | 0.479 | 0.126 |
| 6 | 4 | 8.158 | 0.026 | 0.165 | 0.470 | 0.124 |
| 6 | 8 | 7.489 | 0.085 | 0.268 | 0.567 | 0.294 |
| 6 | 16 | 7.368 | 0.102 | 0.299 | 0.582 | 0.333 |
| 6 | 32 | 7.349 | 0.112 | 0.296 | 0.579 | 0.343 |
| 6 | 64 | 7.496 | 0.081 | 0.257 | 0.575 | 0.293 |
| 6 | 128 | 7.467 | 0.086 | 0.264 | 0.580 | 0.293 |

*Table C.12: Validation Metrics for N = 5 (Indistinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.151 | 0.025 | 0.179 | 0.484 | 0.124 |
| 1 | 4 | 8.191 | 0.016 | 0.167 | 0.456 | 0.114 |
| 1 | 8 | 7.981 | 0.031 | 0.203 | 0.541 | 0.146 |
| 1 | 16 | 7.916 | 0.035 | 0.197 | 0.549 | 0.159 |
| 1 | 32 | 8.006 | 0.038 | 0.186 | 0.534 | 0.151 |
| 1 | 64 | 8.083 | 0.032 | 0.186 | 0.525 | 0.146 |
| 1 | 128 | 8.106 | 0.027 | 0.189 | 0.504 | 0.134 |

*Continued on next page*

*Table C.12 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 2 | 2 | 8.090 | 0.021 | 0.179 | 0.487 | 0.125 |
| 2 | 4 | 8.138 | 0.024 | 0.172 | 0.461 | 0.126 |
| 2 | 8 | 7.805 | 0.034 | 0.201 | 0.559 | 0.151 |
| 2 | 16 | 7.981 | 0.032 | 0.196 | 0.519 | 0.143 |
| 2 | 32 | 8.030 | 0.030 | 0.196 | 0.534 | 0.149 |
| 2 | 64 | 8.053 | 0.028 | 0.194 | 0.497 | 0.144 |
| 2 | 128 | 8.176 | 0.031 | 0.198 | 0.515 | 0.134 |
| 3 | 2 | 8.152 | 0.023 | 0.175 | 0.482 | 0.125 |
| 3 | 4 | 8.151 | 0.023 | 0.175 | 0.482 | 0.125 |
| 3 | 8 | 8.174 | 0.024 | 0.172 | 0.486 | 0.117 |
| 3 | 16 | 7.898 | 0.033 | 0.199 | 0.557 | 0.154 |
| 3 | 32 | 8.084 | 0.027 | 0.189 | 0.524 | 0.147 |
| 3 | 64 | 8.090 | 0.037 | 0.205 | 0.517 | 0.156 |
| 3 | 128 | 8.372 | 0.029 | 0.170 | 0.490 | 0.132 |
| 4 | 2 | 8.160 | 0.030 | 0.184 | 0.473 | 0.125 |
| 4 | 4 | 8.155 | 0.017 | 0.166 | 0.471 | 0.118 |
| 4 | 8 | 7.859 | 0.038 | 0.211 | 0.556 | 0.146 |
| 4 | 16 | 7.849 | 0.034 | 0.195 | 0.544 | 0.150 |
| 4 | 32 | 7.983 | 0.032 | 0.188 | 0.535 | 0.143 |
| 4 | 64 | 7.925 | 0.037 | 0.210 | 0.546 | 0.148 |
| 4 | 128 | 8.255 | 0.026 | 0.179 | 0.481 | 0.133 |
| 5 | 2 | 8.163 | 0.027 | 0.164 | 0.458 | 0.118 |
| 5 | 4 | 8.085 | 0.019 | 0.168 | 0.477 | 0.134 |
| 5 | 8 | 7.835 | 0.037 | 0.194 | 0.548 | 0.147 |
| 5 | 16 | 7.917 | 0.030 | 0.189 | 0.518 | 0.147 |
| 5 | 32 | 7.875 | 0.038 | 0.191 | 0.538 | 0.161 |
| 5 | 64 | 7.866 | 0.034 | 0.214 | 0.553 | 0.149 |
| 5 | 128 | 8.116 | 0.025 | 0.186 | 0.518 | 0.142 |
| 6 | 2 | 8.151 | 0.023 | 0.175 | 0.482 | 0.125 |
| 6 | 4 | 8.154 | 0.022 | 0.179 | 0.482 | 0.129 |
| 6 | 8 | 7.993 | 0.032 | 0.186 | 0.508 | 0.141 |
| 6 | 16 | 7.918 | 0.037 | 0.211 | 0.545 | 0.168 |
| 6 | 32 | 7.950 | 0.031 | 0.195 | 0.522 | 0.153 |
| 6 | 64 | 7.839 | 0.034 | 0.202 | 0.547 | 0.155 |
| 6 | 128 | 7.912 | 0.030 | 0.203 | 0.547 | 0.153 |

*Table C.13: Model Metrics for N = 6 (Indistinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 84 | 19 | 105.2 | 20,192 |
| 1 | 4 | 84 | 40 | 277.1 | 29,428 |
| 1 | 8 | 84 | 14 | 139.1 | 47,924 |
| 1 | 16 | 84 | 14 | 35.0 | 85,012 |
| 1 | 32 | 84 | 20 | 44.4 | 159,572 |
| 1 | 64 | 84 | 21 | 55.9 | 310,228 |
| 1 | 128 | 84 | 18 | 73.2 | 617,684 |
| 2 | 2 | 84 | 16 | 8.3 | 20,198 |
| 2 | 4 | 84 | 81 | 40.2 | 29,448 |
| 2 | 8 | 84 | 30 | 17.2 | 47,996 |
| 2 | 16 | 84 | 23 | 55.1 | 85,284 |
| 2 | 32 | 84 | 20 | 53.3 | 160,628 |
| 2 | 64 | 84 | 20 | 62.7 | 314,388 |
| 2 | 128 | 84 | 16 | 86.3 | 634,196 |
| 3 | 2 | 84 | 41 | 26.7 | 20,204 |
| 3 | 4 | 84 | 33 | 21.7 | 29,468 |
| 3 | 8 | 84 | 12 | 11.6 | 48,068 |
| 3 | 16 | 84 | 22 | 62.9 | 85,556 |
| 3 | 32 | 84 | 21 | 62.0 | 161,684 |
| 3 | 64 | 84 | 18 | 76.7 | 318,548 |
| 3 | 128 | 84 | 16 | 104.7 | 650,708 |
| 4 | 2 | 84 | 15 | 12.9 | 20,210 |
| 4 | 4 | 84 | 40 | 27.3 | 29,488 |
| 4 | 8 | 84 | 13 | 11.9 | 48,140 |
| 4 | 16 | 84 | 24 | 86.5 | 85,828 |
| 4 | 32 | 84 | 24 | 85.3 | 162,740 |
| 4 | 64 | 84 | 21 | 96.2 | 322,708 |
| 4 | 128 | 84 | 17 | 123.5 | 667,220 |
| 5 | 2 | 84 | 98 | 66.5 | 20,216 |
| 5 | 4 | 84 | 13 | 11.6 | 29,508 |
| 5 | 8 | 84 | 19 | 17.5 | 48,212 |
| 5 | 16 | 84 | 25 | 60.7 | 86,100 |
| 5 | 32 | 84 | 24 | 67.5 | 163,796 |
| 5 | 64 | 84 | 20 | 76.8 | 326,868 |
| 5 | 128 | 84 | 20 | 151.2 | 683,732 |

*Continued on next page*

*Table C.13 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 6 | 2 | 84 | 86 | 61.7 | 20,222 |
| 6 | 4 | 84 | 23 | 20.5 | 29,528 |
| 6 | 8 | 84 | 42 | 39.7 | 48,284 |
| 6 | 16 | 84 | 27 | 77.3 | 86,372 |
| 6 | 32 | 84 | 25 | 80.8 | 164,852 |
| 6 | 64 | 84 | 20 | 91.6 | 331,028 |
| 6 | 128 | 84 | 20 | 171.5 | 700,244 |

*Table C.14: Training Metrics for N = 6 (Indistinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.151 | 0.039 | 0.183 | 0.458 | 0.163 |
| 1 | 4 | 7.608 | 0.090 | 0.252 | 0.520 | 0.295 |
| 1 | 8 | 7.964 | 0.065 | 0.204 | 0.471 | 0.221 |
| 1 | 16 | 7.605 | 0.118 | 0.272 | 0.503 | 0.346 |
| 1 | 32 | 7.095 | 0.171 | 0.349 | 0.595 | 0.446 |
| 1 | 64 | 6.894 | 0.203 | 0.393 | 0.627 | 0.497 |
| 1 | 128 | 6.681 | 0.260 | 0.434 | 0.625 | 0.571 |
| 2 | 2 | 8.200 | 0.025 | 0.170 | 0.446 | 0.126 |
| 2 | 4 | 7.537 | 0.087 | 0.259 | 0.544 | 0.290 |
| 2 | 8 | 7.296 | 0.128 | 0.308 | 0.575 | 0.383 |
| 2 | 16 | 7.215 | 0.145 | 0.322 | 0.576 | 0.403 |
| 2 | 32 | 6.835 | 0.213 | 0.405 | 0.622 | 0.513 |
| 2 | 64 | 6.768 | 0.220 | 0.416 | 0.622 | 0.533 |
| 2 | 128 | 6.874 | 0.202 | 0.397 | 0.610 | 0.497 |
| 3 | 2 | 8.199 | 0.027 | 0.171 | 0.448 | 0.125 |
| 3 | 4 | 7.771 | 0.075 | 0.230 | 0.495 | 0.267 |
| 3 | 8 | 8.084 | 0.051 | 0.190 | 0.457 | 0.191 |
| 3 | 16 | 7.163 | 0.160 | 0.344 | 0.582 | 0.433 |
| 3 | 32 | 6.955 | 0.193 | 0.379 | 0.600 | 0.485 |
| 3 | 64 | 6.519 | 0.281 | 0.470 | 0.654 | 0.613 |
| 3 | 128 | 6.646 | 0.243 | 0.446 | 0.641 | 0.565 |
| 4 | 2 | 8.044 | 0.053 | 0.194 | 0.445 | 0.196 |
| 4 | 4 | 7.456 | 0.117 | 0.293 | 0.555 | 0.348 |
| 4 | 8 | 8.170 | 0.033 | 0.170 | 0.448 | 0.147 |

*Continued on next page*

*Table C.14 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 4 | 16 | 7.476 | 0.098 | 0.273 | 0.557 | 0.316 |
| 4 | 32 | 7.218 | 0.148 | 0.330 | 0.577 | 0.408 |
| 4 | 64 | 7.144 | 0.156 | 0.349 | 0.594 | 0.423 |
| 4 | 128 | 6.504 | 0.272 | 0.484 | 0.668 | 0.604 |
| 5 | 2 | 8.200 | 0.027 | 0.167 | 0.461 | 0.125 |
| 5 | 4 | 8.201 | 0.027 | 0.167 | 0.461 | 0.122 |
| 5 | 8 | 8.183 | 0.033 | 0.169 | 0.444 | 0.149 |
| 5 | 16 | 7.568 | 0.081 | 0.252 | 0.554 | 0.274 |
| 5 | 32 | 7.341 | 0.115 | 0.290 | 0.556 | 0.361 |
| 5 | 64 | 7.286 | 0.124 | 0.307 | 0.572 | 0.371 |
| 5 | 128 | 7.175 | 0.150 | 0.339 | 0.587 | 0.406 |
| 6 | 2 | 8.200 | 0.027 | 0.167 | 0.461 | 0.125 |
| 6 | 4 | 8.200 | 0.025 | 0.170 | 0.446 | 0.125 |
| 6 | 8 | 7.632 | 0.080 | 0.238 | 0.536 | 0.268 |
| 6 | 16 | 7.550 | 0.087 | 0.258 | 0.552 | 0.283 |
| 6 | 32 | 7.362 | 0.109 | 0.293 | 0.573 | 0.339 |
| 6 | 64 | 7.308 | 0.127 | 0.310 | 0.569 | 0.361 |
| 6 | 128 | 7.290 | 0.124 | 0.303 | 0.575 | 0.375 |

*Table C.15: Validation Metrics for N = 6 (Indistinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.212 | 0.028 | 0.153 | 0.444 | 0.123 |
| 1 | 4 | 8.110 | 0.023 | 0.172 | 0.496 | 0.130 |
| 1 | 8 | 8.307 | 0.024 | 0.170 | 0.452 | 0.111 |
| 1 | 16 | 8.414 | 0.020 | 0.151 | 0.436 | 0.111 |
| 1 | 32 | 8.254 | 0.027 | 0.180 | 0.501 | 0.127 |
| 1 | 64 | 8.417 | 0.027 | 0.169 | 0.505 | 0.130 |
| 1 | 128 | 8.658 | 0.029 | 0.162 | 0.477 | 0.131 |
| 2 | 2 | 8.210 | 0.016 | 0.156 | 0.442 | 0.113 |
| 2 | 4 | 7.946 | 0.031 | 0.199 | 0.518 | 0.146 |
| 2 | 8 | 8.178 | 0.023 | 0.183 | 0.497 | 0.125 |
| 2 | 16 | 8.177 | 0.023 | 0.177 | 0.497 | 0.118 |
| 2 | 32 | 8.570 | 0.023 | 0.174 | 0.471 | 0.115 |
| 2 | 64 | 8.626 | 0.027 | 0.168 | 0.489 | 0.125 |

*Continued on next page*

*Table C.15 – continued from previous page*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 2 | 128 | 8.453 | 0.026 | 0.164 | 0.485 | 0.117 |
| 3 | 2 | 8.204 | 0.020 | 0.158 | 0.441 | 0.114 |
| 3 | 4 | 8.168 | 0.024 | 0.169 | 0.467 | 0.120 |
| 3 | 8 | 8.276 | 0.017 | 0.148 | 0.430 | 0.106 |
| 3 | 16 | 8.254 | 0.029 | 0.189 | 0.504 | 0.136 |
| 3 | 32 | 8.357 | 0.026 | 0.176 | 0.494 | 0.124 |
| 3 | 64 | 8.787 | 0.027 | 0.164 | 0.468 | 0.110 |
| 3 | 128 | 8.608 | 0.024 | 0.166 | 0.470 | 0.119 |
| 4 | 2 | 8.288 | 0.024 | 0.155 | 0.429 | 0.116 |
| 4 | 4 | 8.168 | 0.023 | 0.183 | 0.490 | 0.136 |
| 4 | 8 | 8.208 | 0.021 | 0.157 | 0.446 | 0.108 |
| 4 | 16 | 7.982 | 0.024 | 0.176 | 0.524 | 0.134 |
| 4 | 32 | 8.162 | 0.024 | 0.183 | 0.505 | 0.130 |
| 4 | 64 | 8.180 | 0.030 | 0.182 | 0.493 | 0.138 |
| 4 | 128 | 8.713 | 0.023 | 0.169 | 0.492 | 0.121 |
| 5 | 2 | 8.205 | 0.024 | 0.166 | 0.450 | 0.114 |
| 5 | 4 | 8.207 | 0.024 | 0.166 | 0.450 | 0.120 |
| 5 | 8 | 8.208 | 0.029 | 0.173 | 0.442 | 0.111 |
| 5 | 16 | 7.932 | 0.031 | 0.189 | 0.526 | 0.147 |
| 5 | 32 | 8.154 | 0.024 | 0.182 | 0.520 | 0.135 |
| 5 | 64 | 8.114 | 0.025 | 0.184 | 0.507 | 0.132 |
| 5 | 128 | 8.301 | 0.033 | 0.177 | 0.484 | 0.125 |
| 6 | 2 | 8.205 | 0.024 | 0.166 | 0.450 | 0.114 |
| 6 | 4 | 8.206 | 0.016 | 0.156 | 0.442 | 0.114 |
| 6 | 8 | 7.964 | 0.035 | 0.193 | 0.516 | 0.148 |
| 6 | 16 | 8.036 | 0.027 | 0.185 | 0.509 | 0.148 |
| 6 | 32 | 8.083 | 0.033 | 0.186 | 0.513 | 0.137 |
| 6 | 64 | 8.071 | 0.031 | 0.205 | 0.505 | 0.143 |
| 6 | 128 | 8.146 | 0.027 | 0.176 | 0.498 | 0.141 |

*Table C.16: Model Metrics for N = 7 (Indistinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 85 | 12 | 8.2 | 23,649 |
| 1 | 4 | 85 | 15 | 10.3 | 36,213 |

*Continued on next page*

*Table C.16 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 8 | 85 | 27 | 17.9 | 61,365 |
| 1 | 16 | 85 | 11 | 8.6 | 111,765 |
| 1 | 32 | 85 | 13 | 13.3 | 212,949 |
| 1 | 64 | 85 | 19 | 36.3 | 416,853 |
| 1 | 128 | 85 | 13 | 46.2 | 830,805 |
| 2 | 2 | 85 | 16 | 11.2 | 23,655 |
| 2 | 4 | 85 | 14 | 11.0 | 36,233 |
| 2 | 8 | 85 | 12 | 10.7 | 61,437 |
| 2 | 16 | 85 | 11 | 10.2 | 112,037 |
| 2 | 32 | 85 | 11 | 13.9 | 214,005 |
| 2 | 64 | 85 | 18 | 43.5 | 421,013 |
| 2 | 128 | 85 | 16 | 75.8 | 847,317 |
| 3 | 2 | 85 | 14 | 11.8 | 23,661 |
| 3 | 4 | 85 | 22 | 17.7 | 36,253 |
| 3 | 8 | 85 | 28 | 24.7 | 61,509 |
| 3 | 16 | 85 | 12 | 14.7 | 112,309 |
| 3 | 32 | 85 | 20 | 32.5 | 215,061 |
| 3 | 64 | 85 | 19 | 60.9 | 425,173 |
| 3 | 128 | 85 | 18 | 115.8 | 863,829 |
| 4 | 2 | 85 | 16 | 14.0 | 23,667 |
| 4 | 4 | 85 | 16 | 15.7 | 36,273 |
| 4 | 8 | 85 | 21 | 21.8 | 61,581 |
| 4 | 16 | 85 | 25 | 32.1 | 112,581 |
| 4 | 32 | 85 | 21 | 41.1 | 216,117 |
| 4 | 64 | 85 | 20 | 74.9 | 429,333 |
| 4 | 128 | 85 | 17 | 129.5 | 880,341 |
| 5 | 2 | 85 | 91 | 72.0 | 23,673 |
| 5 | 4 | 85 | 15 | 17.1 | 36,293 |
| 5 | 8 | 85 | 30 | 34.5 | 61,653 |
| 5 | 16 | 85 | 28 | 41.2 | 112,853 |
| 5 | 32 | 85 | 21 | 45.9 | 217,173 |
| 5 | 64 | 85 | 14 | 61.7 | 433,493 |
| 5 | 128 | 85 | 21 | 184.8 | 896,853 |
| 6 | 2 | 85 | 82 | 73.8 | 23,679 |
| 6 | 4 | 85 | 17 | 21.1 | 36,313 |
| 6 | 8 | 85 | 17 | 24.6 | 61,725 |

*Continued on next page*

*Table C.16 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 6 | 16 | 85 | 25 | 43.5 | 113,125 |
| 6 | 32 | 85 | 25 | 63.5 | 218,229 |
| 6 | 64 | 85 | 19 | 95.4 | 437,653 |
| 6 | 128 | 85 | 23 | 226.0 | 913,365 |

*Table C.17: Training Metrics for N = 7 (Indistinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.101 | 0.051 | 0.198 | 0.457 | 0.199 |
| 1 | 4 | 8.003 | 0.062 | 0.209 | 0.471 | 0.230 |
| 1 | 8 | 6.973 | 0.203 | 0.380 | 0.599 | 0.494 |
| 1 | 16 | 7.724 | 0.102 | 0.256 | 0.497 | 0.319 |
| 1 | 32 | 7.406 | 0.141 | 0.305 | 0.538 | 0.380 |
| 1 | 64 | 6.557 | 0.280 | 0.471 | 0.661 | 0.593 |
| 1 | 128 | 7.156 | 0.180 | 0.347 | 0.552 | 0.459 |
| 2 | 2 | 8.214 | 0.025 | 0.172 | 0.471 | 0.130 |
| 2 | 4 | 8.145 | 0.041 | 0.183 | 0.470 | 0.173 |
| 2 | 8 | 7.961 | 0.065 | 0.221 | 0.485 | 0.243 |
| 2 | 16 | 7.891 | 0.074 | 0.226 | 0.490 | 0.246 |
| 2 | 32 | 7.593 | 0.120 | 0.274 | 0.502 | 0.353 |
| 2 | 64 | 6.155 | 0.346 | 0.550 | 0.716 | 0.684 |
| 2 | 128 | 6.106 | 0.351 | 0.563 | 0.701 | 0.692 |
| 3 | 2 | 8.212 | 0.031 | 0.173 | 0.476 | 0.130 |
| 3 | 4 | 8.088 | 0.050 | 0.196 | 0.469 | 0.195 |
| 3 | 8 | 7.059 | 0.182 | 0.360 | 0.584 | 0.463 |
| 3 | 16 | 7.837 | 0.088 | 0.237 | 0.481 | 0.281 |
| 3 | 32 | 6.315 | 0.297 | 0.521 | 0.676 | 0.634 |
| 3 | 64 | 5.718 | 0.421 | 0.643 | 0.764 | 0.763 |
| 3 | 128 | 6.058 | 0.350 | 0.574 | 0.713 | 0.698 |
| 4 | 2 | 8.216 | 0.027 | 0.168 | 0.473 | 0.130 |
| 4 | 4 | 8.214 | 0.030 | 0.172 | 0.474 | 0.130 |
| 4 | 8 | 8.210 | 0.027 | 0.168 | 0.470 | 0.131 |
| 4 | 16 | 7.052 | 0.166 | 0.365 | 0.596 | 0.443 |
| 4 | 32 | 7.286 | 0.134 | 0.315 | 0.561 | 0.379 |
| 4 | 64 | 6.803 | 0.213 | 0.407 | 0.619 | 0.519 |

*Continued on next page*

*Table C.17 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 4 | 128 | 5.374 | 0.463 | 0.713 | 0.803 | 0.807 |
| 5 | 2 | 8.213 | 0.027 | 0.168 | 0.473 | 0.130 |
| 5 | 4 | 8.216 | 0.027 | 0.168 | 0.473 | 0.124 |
| 5 | 8 | 7.443 | 0.111 | 0.287 | 0.536 | 0.337 |
| 5 | 16 | 7.077 | 0.160 | 0.356 | 0.594 | 0.434 |
| 5 | 32 | 6.367 | 0.303 | 0.509 | 0.674 | 0.634 |
| 5 | 64 | 6.828 | 0.232 | 0.415 | 0.615 | 0.545 |
| 5 | 128 | 7.007 | 0.161 | 0.377 | 0.606 | 0.441 |
| 6 | 2 | 8.213 | 0.027 | 0.168 | 0.473 | 0.130 |
| 6 | 4 | 8.210 | 0.027 | 0.172 | 0.472 | 0.130 |
| 6 | 8 | 8.116 | 0.046 | 0.193 | 0.463 | 0.188 |
| 6 | 16 | 7.241 | 0.141 | 0.332 | 0.577 | 0.398 |
| 6 | 32 | 7.277 | 0.129 | 0.319 | 0.571 | 0.371 |
| 6 | 64 | 7.044 | 0.179 | 0.368 | 0.590 | 0.461 |
| 6 | 128 | 6.229 | 0.290 | 0.532 | 0.694 | 0.630 |

*Table C.18: Validation Metrics for N = 7 (Indistinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.259 | 0.020 | 0.164 | 0.445 | 0.115 |
| 1 | 4 | 8.280 | 0.026 | 0.155 | 0.439 | 0.115 |
| 1 | 8 | 8.397 | 0.026 | 0.180 | 0.500 | 0.132 |
| 1 | 16 | 8.387 | 0.023 | 0.160 | 0.439 | 0.117 |
| 1 | 32 | 8.408 | 0.032 | 0.171 | 0.470 | 0.139 |
| 1 | 64 | 8.631 | 0.026 | 0.181 | 0.478 | 0.140 |
| 1 | 128 | 8.587 | 0.023 | 0.152 | 0.439 | 0.116 |
| 2 | 2 | 8.223 | 0.033 | 0.171 | 0.465 | 0.126 |
| 2 | 4 | 8.234 | 0.021 | 0.167 | 0.463 | 0.116 |
| 2 | 8 | 8.230 | 0.026 | 0.161 | 0.447 | 0.110 |
| 2 | 16 | 8.289 | 0.024 | 0.167 | 0.459 | 0.111 |
| 2 | 32 | 8.492 | 0.020 | 0.161 | 0.441 | 0.112 |
| 2 | 64 | 9.041 | 0.027 | 0.166 | 0.456 | 0.123 |
| 2 | 128 | 9.158 | 0.025 | 0.162 | 0.462 | 0.126 |
| 3 | 2 | 8.224 | 0.028 | 0.169 | 0.459 | 0.118 |
| 3 | 4 | 8.270 | 0.024 | 0.164 | 0.441 | 0.116 |

*Continued on next page*

*Table C.18 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 3 | 8 | 8.308 | 0.022 | 0.176 | 0.481 | 0.121 |
| 3 | 16 | 8.367 | 0.017 | 0.147 | 0.446 | 0.116 |
| 3 | 32 | 8.900 | 0.027 | 0.168 | 0.461 | 0.122 |
| 3 | 64 | 9.536 | 0.025 | 0.161 | 0.452 | 0.114 |
| 3 | 128 | 9.077 | 0.025 | 0.149 | 0.444 | 0.118 |
| 4 | 2 | 8.222 | 0.023 | 0.171 | 0.462 | 0.126 |
| 4 | 4 | 8.222 | 0.020 | 0.168 | 0.454 | 0.123 |
| 4 | 8 | 8.222 | 0.023 | 0.163 | 0.459 | 0.130 |
| 4 | 16 | 8.353 | 0.025 | 0.185 | 0.484 | 0.134 |
| 4 | 32 | 8.239 | 0.023 | 0.165 | 0.485 | 0.128 |
| 4 | 64 | 8.560 | 0.025 | 0.154 | 0.474 | 0.125 |
| 4 | 128 | 9.717 | 0.032 | 0.169 | 0.459 | 0.119 |
| 5 | 2 | 8.224 | 0.023 | 0.171 | 0.462 | 0.126 |
| 5 | 4 | 8.223 | 0.023 | 0.171 | 0.462 | 0.122 |
| 5 | 8 | 8.215 | 0.022 | 0.178 | 0.482 | 0.129 |
| 5 | 16 | 8.387 | 0.030 | 0.195 | 0.496 | 0.133 |
| 5 | 32 | 8.796 | 0.025 | 0.161 | 0.450 | 0.124 |
| 5 | 64 | 8.597 | 0.024 | 0.154 | 0.446 | 0.115 |
| 5 | 128 | 8.373 | 0.028 | 0.181 | 0.495 | 0.130 |
| 6 | 2 | 8.223 | 0.023 | 0.171 | 0.462 | 0.126 |
| 6 | 4 | 8.224 | 0.031 | 0.168 | 0.464 | 0.117 |
| 6 | 8 | 8.268 | 0.026 | 0.155 | 0.449 | 0.122 |
| 6 | 16 | 8.281 | 0.027 | 0.175 | 0.485 | 0.122 |
| 6 | 32 | 8.248 | 0.028 | 0.178 | 0.481 | 0.129 |
| 6 | 64 | 8.362 | 0.028 | 0.182 | 0.497 | 0.132 |
| 6 | 128 | 9.011 | 0.019 | 0.161 | 0.476 | 0.129 |

*Table C.19: Model Metrics for N = 8 (Indistinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 84 | 14 | 20.8 | 27,360 |
| 1 | 4 | 84 | 15 | 26.7 | 43,764 |
| 1 | 8 | 84 | 14 | 26.2 | 76,596 |
| 1 | 16 | 84 | 13 | 14.4 | 142,356 |
| 1 | 32 | 84 | 12 | 20.2 | 274,260 |

*Continued on next page*

*Table C.19 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 64 | 84 | 12 | 34.2 | 539,604 |
| 1 | 128 | 84 | 12 | 60.4 | 1,076,436 |
| 2 | 2 | 84 | 15 | 25.2 | 27,366 |
| 2 | 4 | 84 | 12 | 25.5 | 43,784 |
| 2 | 8 | 84 | 13 | 28.5 | 76,668 |
| 2 | 16 | 84 | 11 | 14.9 | 142,628 |
| 2 | 32 | 84 | 12 | 24.0 | 275,316 |
| 2 | 64 | 84 | 11 | 37.8 | 543,764 |
| 2 | 128 | 84 | 11 | 70.8 | 1,092,948 |
| 3 | 2 | 84 | 20 | 33.4 | 27,372 |
| 3 | 4 | 84 | 15 | 34.2 | 43,804 |
| 3 | 8 | 84 | 18 | 42.0 | 76,740 |
| 3 | 16 | 84 | 14 | 29.5 | 142,900 |
| 3 | 32 | 84 | 11 | 32.9 | 276,372 |
| 3 | 64 | 84 | 11 | 49.6 | 547,924 |
| 3 | 128 | 84 | 13 | 111.2 | 1,109,460 |
| 4 | 2 | 84 | 68 | 113.1 | 27,378 |
| 4 | 4 | 84 | 17 | 42.7 | 43,824 |
| 4 | 8 | 84 | 16 | 43.1 | 76,812 |
| 4 | 16 | 84 | 14 | 37.0 | 143,172 |
| 4 | 32 | 84 | 16 | 53.0 | 277,428 |
| 4 | 64 | 84 | 12 | 63.7 | 552,084 |
| 4 | 128 | 84 | 13 | 127.6 | 1,125,972 |
| 5 | 2 | 84 | 14 | 31.4 | 27,384 |
| 5 | 4 | 84 | 25 | 68.7 | 43,844 |
| 5 | 8 | 84 | 13 | 40.3 | 76,884 |
| 5 | 16 | 84 | 14 | 42.9 | 143,444 |
| 5 | 32 | 84 | 12 | 46.1 | 278,484 |
| 5 | 64 | 84 | 18 | 106.4 | 556,244 |
| 5 | 128 | 84 | 14 | 157.3 | 1,142,484 |
| 6 | 2 | 84 | 63 | 125.8 | 27,390 |
| 6 | 4 | 84 | 28 | 85.0 | 43,864 |
| 6 | 8 | 84 | 15 | 50.4 | 76,956 |
| 6 | 16 | 84 | 26 | 84.1 | 143,716 |
| 6 | 32 | 84 | 22 | 90.5 | 279,540 |
| 6 | 64 | 84 | 21 | 135.0 | 560,404 |

*Continued on next page*

*Table C.19 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 6 | 128 | 84 | 20 | 249.8 | 1,158,996 |

*Table C.20: Training Metrics for N = 8 (Indistinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.125 | 0.044 | 0.181 | 0.446 | 0.182 |
| 1 | 4 | 7.980 | 0.066 | 0.206 | 0.474 | 0.222 |
| 1 | 8 | 7.765 | 0.097 | 0.249 | 0.495 | 0.302 |
| 1 | 16 | 7.590 | 0.137 | 0.271 | 0.486 | 0.375 |
| 1 | 32 | 7.370 | 0.166 | 0.313 | 0.519 | 0.435 |
| 1 | 64 | 7.134 | 0.192 | 0.359 | 0.575 | 0.477 |
| 1 | 128 | 6.842 | 0.235 | 0.416 | 0.624 | 0.540 |
| 2 | 2 | 7.970 | 0.065 | 0.199 | 0.460 | 0.227 |
| 2 | 4 | 8.062 | 0.060 | 0.197 | 0.455 | 0.206 |
| 2 | 8 | 7.974 | 0.074 | 0.210 | 0.461 | 0.237 |
| 2 | 16 | 7.672 | 0.120 | 0.263 | 0.485 | 0.350 |
| 2 | 32 | 7.055 | 0.197 | 0.358 | 0.591 | 0.495 |
| 2 | 64 | 6.923 | 0.217 | 0.395 | 0.595 | 0.526 |
| 2 | 128 | 6.172 | 0.381 | 0.560 | 0.704 | 0.714 |
| 3 | 2 | 8.194 | 0.028 | 0.168 | 0.464 | 0.142 |
| 3 | 4 | 8.043 | 0.059 | 0.195 | 0.461 | 0.203 |
| 3 | 8 | 7.410 | 0.157 | 0.317 | 0.521 | 0.422 |
| 3 | 16 | 7.492 | 0.142 | 0.298 | 0.513 | 0.392 |
| 3 | 32 | 7.405 | 0.153 | 0.302 | 0.534 | 0.408 |
| 3 | 64 | 6.379 | 0.354 | 0.518 | 0.658 | 0.680 |
| 3 | 128 | 5.938 | 0.398 | 0.615 | 0.727 | 0.748 |
| 4 | 2 | 8.211 | 0.025 | 0.166 | 0.461 | 0.121 |
| 4 | 4 | 8.212 | 0.027 | 0.166 | 0.464 | 0.120 |
| 4 | 8 | 7.930 | 0.072 | 0.211 | 0.459 | 0.252 |
| 4 | 16 | 7.289 | 0.180 | 0.339 | 0.528 | 0.460 |
| 4 | 32 | 7.455 | 0.128 | 0.292 | 0.541 | 0.373 |
| 4 | 64 | 7.051 | 0.207 | 0.373 | 0.573 | 0.507 |
| 4 | 128 | 6.314 | 0.331 | 0.527 | 0.680 | 0.658 |
| 5 | 2 | 8.213 | 0.026 | 0.160 | 0.456 | 0.117 |
| 5 | 4 | 7.597 | 0.112 | 0.271 | 0.510 | 0.333 |

*Continued on next page*

*Table C.20 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 5 | 8 | 7.892 | 0.070 | 0.220 | 0.475 | 0.251 |
| 5 | 16 | 7.843 | 0.083 | 0.226 | 0.479 | 0.275 |
| 5 | 32 | 7.562 | 0.122 | 0.288 | 0.520 | 0.354 |
| 5 | 64 | 5.022 | 0.542 | 0.778 | 0.849 | 0.870 |
| 5 | 128 | 7.015 | 0.189 | 0.377 | 0.598 | 0.475 |
| 6 | 2 | 8.211 | 0.025 | 0.167 | 0.462 | 0.121 |
| 6 | 4 | 8.157 | 0.043 | 0.175 | 0.442 | 0.169 |
| 6 | 8 | 8.130 | 0.042 | 0.171 | 0.448 | 0.175 |
| 6 | 16 | 7.262 | 0.143 | 0.323 | 0.568 | 0.395 |
| 6 | 32 | 6.273 | 0.290 | 0.538 | 0.688 | 0.628 |
| 6 | 64 | 5.818 | 0.360 | 0.615 | 0.745 | 0.721 |
| 6 | 128 | 6.404 | 0.264 | 0.500 | 0.653 | 0.607 |

*Table C.21: Validation Metrics for N = 8 (Indistinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.250 | 0.027 | 0.159 | 0.439 | 0.117 |
| 1 | 4 | 8.268 | 0.023 | 0.163 | 0.463 | 0.117 |
| 1 | 8 | 8.291 | 0.026 | 0.169 | 0.454 | 0.125 |
| 1 | 16 | 8.527 | 0.025 | 0.146 | 0.410 | 0.101 |
| 1 | 32 | 8.578 | 0.024 | 0.149 | 0.412 | 0.116 |
| 1 | 64 | 8.524 | 0.021 | 0.168 | 0.466 | 0.113 |
| 1 | 128 | 8.562 | 0.027 | 0.164 | 0.462 | 0.111 |
| 2 | 2 | 8.322 | 0.029 | 0.164 | 0.451 | 0.109 |
| 2 | 4 | 8.294 | 0.016 | 0.153 | 0.433 | 0.106 |
| 2 | 8 | 8.322 | 0.021 | 0.152 | 0.432 | 0.104 |
| 2 | 16 | 8.468 | 0.023 | 0.147 | 0.423 | 0.110 |
| 2 | 32 | 8.343 | 0.028 | 0.173 | 0.476 | 0.138 |
| 2 | 64 | 8.581 | 0.028 | 0.172 | 0.463 | 0.116 |
| 2 | 128 | 8.890 | 0.022 | 0.161 | 0.454 | 0.117 |
| 3 | 2 | 8.213 | 0.028 | 0.161 | 0.473 | 0.121 |
| 3 | 4 | 8.299 | 0.026 | 0.156 | 0.444 | 0.121 |
| 3 | 8 | 8.614 | 0.020 | 0.154 | 0.403 | 0.110 |
| 3 | 16 | 8.601 | 0.020 | 0.136 | 0.418 | 0.100 |
| 3 | 32 | 8.584 | 0.025 | 0.155 | 0.443 | 0.113 |

*Continued on next page*

*Table C.21 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 3 | 64 | 9.062 | 0.023 | 0.147 | 0.431 | 0.106 |
| 3 | 128 | 9.162 | 0.025 | 0.145 | 0.427 | 0.118 |
| 4 | 2 | 8.210 | 0.026 | 0.165 | 0.464 | 0.123 |
| 4 | 4 | 8.212 | 0.017 | 0.166 | 0.467 | 0.117 |
| 4 | 8 | 8.351 | 0.020 | 0.146 | 0.430 | 0.106 |
| 4 | 16 | 8.829 | 0.018 | 0.135 | 0.402 | 0.091 |
| 4 | 32 | 8.373 | 0.027 | 0.168 | 0.476 | 0.108 |
| 4 | 64 | 8.728 | 0.026 | 0.155 | 0.456 | 0.109 |
| 4 | 128 | 9.082 | 0.021 | 0.148 | 0.444 | 0.112 |
| 5 | 2 | 8.213 | 0.026 | 0.171 | 0.462 | 0.119 |
| 5 | 4 | 8.325 | 0.029 | 0.161 | 0.441 | 0.120 |
| 5 | 8 | 8.286 | 0.025 | 0.162 | 0.433 | 0.126 |
| 5 | 16 | 8.347 | 0.021 | 0.155 | 0.437 | 0.101 |
| 5 | 32 | 8.422 | 0.025 | 0.155 | 0.428 | 0.111 |
| 5 | 64 | 10.344 | 0.024 | 0.148 | 0.426 | 0.116 |
| 5 | 128 | 8.470 | 0.025 | 0.144 | 0.454 | 0.113 |
| 6 | 2 | 8.210 | 0.021 | 0.164 | 0.467 | 0.123 |
| 6 | 4 | 8.233 | 0.023 | 0.160 | 0.431 | 0.110 |
| 6 | 8 | 8.261 | 0.025 | 0.168 | 0.426 | 0.117 |
| 6 | 16 | 8.254 | 0.023 | 0.180 | 0.479 | 0.119 |
| 6 | 32 | 9.237 | 0.021 | 0.137 | 0.441 | 0.105 |
| 6 | 64 | 9.503 | 0.018 | 0.152 | 0.445 | 0.114 |
| 6 | 128 | 9.188 | 0.020 | 0.152 | 0.442 | 0.106 |

*Table C.22: Model Metrics for N = 9 (Indistinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 88 | 31 | 77.5 | 32,228 |
| 1 | 4 | 88 | 13 | 42.4 | 52,984 |
| 1 | 8 | 88 | 11 | 39.7 | 94,520 |
| 1 | 16 | 88 | 13 | 19.7 | 177,688 |
| 1 | 32 | 88 | 11 | 24.4 | 344,408 |
| 1 | 64 | 88 | 11 | 38.7 | 679,384 |
| 1 | 128 | 88 | 11 | 70.1 | 1,355,480 |

*Continued on next page*

*Table C.22 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 2 | 2 | 88 | 14 | 44.2 | 32,234 |
| 2 | 4 | 88 | 12 | 43.6 | 53,004 |
| 2 | 8 | 88 | 15 | 55.7 | 94,592 |
| 2 | 16 | 88 | 12 | 20.8 | 177,960 |
| 2 | 32 | 88 | 12 | 30.4 | 345,464 |
| 2 | 64 | 88 | 12 | 50.0 | 683,544 |
| 2 | 128 | 88 | 12 | 95.4 | 1,371,992 |
| 3 | 2 | 88 | 14 | 47.4 | 32,240 |
| 3 | 4 | 88 | 13 | 50.8 | 53,024 |
| 3 | 8 | 88 | 14 | 58.4 | 94,664 |
| 3 | 16 | 88 | 13 | 52.0 | 178,232 |
| 3 | 32 | 88 | 11 | 48.8 | 346,520 |
| 3 | 64 | 88 | 13 | 84.2 | 687,704 |
| 3 | 128 | 88 | 12 | 135.7 | 1,388,504 |
| 4 | 2 | 88 | 56 | 169.7 | 32,246 |
| 4 | 4 | 88 | 18 | 72.8 | 53,044 |
| 4 | 8 | 88 | 21 | 89.5 | 94,736 |
| 4 | 16 | 88 | 12 | 56.4 | 178,504 |
| 4 | 32 | 88 | 13 | 64.6 | 347,576 |
| 4 | 64 | 88 | 13 | 92.7 | 691,864 |
| 4 | 128 | 88 | 13 | 171.8 | 1,405,016 |
| 5 | 2 | 88 | 60 | 198.3 | 32,252 |
| 5 | 4 | 88 | 20 | 84.1 | 53,064 |
| 5 | 8 | 88 | 24 | 123.1 | 94,808 |
| 5 | 16 | 88 | 17 | 80.4 | 178,776 |
| 5 | 32 | 88 | 12 | 71.1 | 348,632 |
| 5 | 64 | 88 | 13 | 114.6 | 696,024 |
| 5 | 128 | 88 | 12 | 190.9 | 1,421,528 |
| 6 | 2 | 88 | 60 | 210.9 | 32,258 |
| 6 | 4 | 88 | 22 | 101.3 | 53,084 |
| 6 | 8 | 88 | 19 | 106.9 | 94,880 |
| 6 | 16 | 88 | 31 | 194.5 | 179,048 |
| 6 | 32 | 88 | 16 | 101.2 | 349,688 |
| 6 | 64 | 88 | 15 | 133.5 | 700,184 |
| 6 | 128 | 88 | 13 | 229.3 | 1,438,040 |

*Table C.23: Training Metrics for N = 9 (Indistinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.175 | 0.045 | 0.184 | 0.468 | 0.178 |
| 1 | 4 | 7.998 | 0.066 | 0.208 | 0.464 | 0.235 |
| 1 | 8 | 7.925 | 0.090 | 0.238 | 0.483 | 0.274 |
| 1 | 16 | 7.692 | 0.116 | 0.269 | 0.482 | 0.350 |
| 1 | 32 | 7.383 | 0.158 | 0.324 | 0.558 | 0.421 |
| 1 | 64 | 7.171 | 0.203 | 0.373 | 0.556 | 0.492 |
| 1 | 128 | 6.473 | 0.326 | 0.515 | 0.667 | 0.659 |
| 2 | 2 | 8.233 | 0.024 | 0.169 | 0.468 | 0.127 |
| 2 | 4 | 8.085 | 0.058 | 0.201 | 0.465 | 0.209 |
| 2 | 8 | 7.458 | 0.159 | 0.305 | 0.515 | 0.408 |
| 2 | 16 | 7.370 | 0.183 | 0.334 | 0.535 | 0.455 |
| 2 | 32 | 7.034 | 0.230 | 0.404 | 0.591 | 0.538 |
| 2 | 64 | 6.270 | 0.368 | 0.547 | 0.709 | 0.698 |
| 2 | 128 | 5.070 | 0.598 | 0.796 | 0.864 | 0.892 |
| 3 | 2 | 8.135 | 0.046 | 0.194 | 0.466 | 0.180 |
| 3 | 4 | 8.028 | 0.069 | 0.212 | 0.466 | 0.233 |
| 3 | 8 | 7.800 | 0.083 | 0.240 | 0.512 | 0.281 |
| 3 | 16 | 7.420 | 0.167 | 0.329 | 0.531 | 0.447 |
| 3 | 32 | 6.984 | 0.222 | 0.393 | 0.598 | 0.523 |
| 3 | 64 | 5.866 | 0.450 | 0.643 | 0.755 | 0.779 |
| 3 | 128 | 4.991 | 0.604 | 0.800 | 0.864 | 0.899 |
| 4 | 2 | 8.234 | 0.027 | 0.166 | 0.470 | 0.127 |
| 4 | 4 | 7.518 | 0.139 | 0.299 | 0.506 | 0.389 |
| 4 | 8 | 7.969 | 0.073 | 0.214 | 0.461 | 0.252 |
| 4 | 16 | 7.619 | 0.115 | 0.271 | 0.516 | 0.348 |
| 4 | 32 | 6.771 | 0.262 | 0.434 | 0.623 | 0.577 |
| 4 | 64 | 5.576 | 0.499 | 0.686 | 0.785 | 0.815 |
| 4 | 128 | 4.893 | 0.605 | 0.815 | 0.873 | 0.908 |
| 5 | 2 | 8.233 | 0.026 | 0.173 | 0.470 | 0.127 |
| 5 | 4 | 8.150 | 0.050 | 0.180 | 0.474 | 0.184 |
| 5 | 8 | 8.060 | 0.060 | 0.203 | 0.468 | 0.214 |
| 5 | 16 | 7.751 | 0.098 | 0.254 | 0.493 | 0.301 |
| 5 | 32 | 7.763 | 0.104 | 0.254 | 0.480 | 0.325 |
| 5 | 64 | 7.009 | 0.219 | 0.391 | 0.584 | 0.521 |
| 5 | 128 | 7.595 | 0.115 | 0.287 | 0.528 | 0.350 |

*Continued on next page*

*Table C.23 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 2 | 8.233 | 0.026 | 0.173 | 0.470 | 0.127 |
| 6 | 4 | 7.983 | 0.060 | 0.207 | 0.483 | 0.217 |
| 6 | 8 | 8.237 | 0.026 | 0.173 | 0.470 | 0.125 |
| 6 | 16 | 6.143 | 0.339 | 0.561 | 0.704 | 0.674 |
| 6 | 32 | 7.670 | 0.100 | 0.261 | 0.510 | 0.309 |
| 6 | 64 | 7.498 | 0.129 | 0.302 | 0.530 | 0.361 |
| 6 | 128 | 7.735 | 0.098 | 0.254 | 0.495 | 0.311 |

*Table C.24: Validation Metrics for N = 9 (Indistinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.276 | 0.025 | 0.159 | 0.441 | 0.111 |
| 1 | 4 | 8.364 | 0.018 | 0.171 | 0.439 | 0.110 |
| 1 | 8 | 8.351 | 0.015 | 0.147 | 0.444 | 0.106 |
| 1 | 16 | 8.576 | 0.023 | 0.131 | 0.397 | 0.099 |
| 1 | 32 | 8.538 | 0.022 | 0.142 | 0.439 | 0.110 |
| 1 | 64 | 8.922 | 0.017 | 0.122 | 0.395 | 0.087 |
| 1 | 128 | 9.130 | 0.021 | 0.142 | 0.432 | 0.107 |
| 2 | 2 | 8.265 | 0.021 | 0.163 | 0.455 | 0.108 |
| 2 | 4 | 8.338 | 0.020 | 0.162 | 0.444 | 0.116 |
| 2 | 8 | 8.752 | 0.019 | 0.145 | 0.418 | 0.102 |
| 2 | 16 | 8.778 | 0.020 | 0.137 | 0.393 | 0.095 |
| 2 | 32 | 8.878 | 0.022 | 0.135 | 0.409 | 0.092 |
| 2 | 64 | 9.184 | 0.023 | 0.156 | 0.435 | 0.110 |
| 2 | 128 | 10.104 | 0.023 | 0.134 | 0.420 | 0.108 |
| 3 | 2 | 8.322 | 0.021 | 0.141 | 0.443 | 0.109 |
| 3 | 4 | 8.411 | 0.023 | 0.152 | 0.442 | 0.101 |
| 3 | 8 | 8.266 | 0.023 | 0.159 | 0.460 | 0.110 |
| 3 | 16 | 8.687 | 0.017 | 0.137 | 0.412 | 0.097 |
| 3 | 32 | 8.711 | 0.020 | 0.144 | 0.438 | 0.106 |
| 3 | 64 | 9.564 | 0.015 | 0.148 | 0.423 | 0.098 |
| 3 | 128 | 10.246 | 0.023 | 0.145 | 0.416 | 0.112 |
| 4 | 2 | 8.265 | 0.020 | 0.163 | 0.455 | 0.107 |
| 4 | 4 | 8.639 | 0.022 | 0.147 | 0.419 | 0.113 |
| 4 | 8 | 8.400 | 0.027 | 0.145 | 0.412 | 0.104 |

*Continued on next page*

*Table C.24 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 4 | 16 | 8.413 | 0.019 | 0.152 | 0.428 | 0.091 |
| 4 | 32 | 8.825 | 0.023 | 0.150 | 0.460 | 0.110 |
| 4 | 64 | 9.789 | 0.025 | 0.151 | 0.444 | 0.108 |
| 4 | 128 | 10.453 | 0.017 | 0.138 | 0.399 | 0.101 |
| 5 | 2 | 8.265 | 0.024 | 0.158 | 0.454 | 0.108 |
| 5 | 4 | 8.275 | 0.021 | 0.149 | 0.445 | 0.115 |
| 5 | 8 | 8.365 | 0.022 | 0.153 | 0.441 | 0.116 |
| 5 | 16 | 8.462 | 0.022 | 0.156 | 0.441 | 0.111 |
| 5 | 32 | 8.531 | 0.019 | 0.137 | 0.433 | 0.102 |
| 5 | 64 | 8.570 | 0.013 | 0.144 | 0.431 | 0.096 |
| 5 | 128 | 8.338 | 0.020 | 0.165 | 0.459 | 0.101 |
| 6 | 2 | 8.265 | 0.024 | 0.158 | 0.454 | 0.108 |
| 6 | 4 | 8.270 | 0.033 | 0.167 | 0.457 | 0.110 |
| 6 | 8 | 8.268 | 0.024 | 0.158 | 0.454 | 0.111 |
| 6 | 16 | 9.322 | 0.020 | 0.155 | 0.438 | 0.125 |
| 6 | 32 | 8.366 | 0.027 | 0.164 | 0.444 | 0.110 |
| 6 | 64 | 8.411 | 0.025 | 0.168 | 0.450 | 0.112 |
| 6 | 128 | 8.389 | 0.019 | 0.161 | 0.450 | 0.112 |

*Table C.25: Model Metrics for N = 10 (Indistinguishable).*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 1 | 2 | 86 | 14 | 76.9 | 36,834 |
| 1 | 4 | 86 | 11 | 65.5 | 62,454 |
| 1 | 8 | 86 | 13 | 83.2 | 113,718 |
| 1 | 16 | 86 | 12 | 28.7 | 216,342 |
| 1 | 32 | 86 | 11 | 32.1 | 421,974 |
| 1 | 64 | 86 | 12 | 57.3 | 834,774 |
| 1 | 128 | 86 | 11 | 91.4 | 1,666,518 |
| 2 | 2 | 86 | 64 | 327.8 | 36,840 |
| 2 | 4 | 86 | 14 | 83.3 | 62,474 |
| 2 | 8 | 86 | 13 | 84.6 | 113,790 |
| 2 | 16 | 86 | 11 | 29.9 | 216,614 |
| 2 | 32 | 86 | 12 | 43.5 | 423,030 |
| 2 | 64 | 86 | 11 | 63.0 | 838,934 |

*Continued on next page*

*Table C.25 – continued from previous page*

| Layers | Filters | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|
| 2 | 128 | 86 | 12 | 127.5 | 1,683,030 |
| 3 | 2 | 86 | 24 | 152.9 | 36,846 |
| 3 | 4 | 86 | 17 | 114.5 | 62,494 |
| 3 | 8 | 86 | 12 | 98.4 | 113,862 |
| 3 | 16 | 86 | 13 | 99.9 | 216,886 |
| 3 | 32 | 86 | 13 | 95.7 | 424,086 |
| 3 | 64 | 86 | 12 | 120.5 | 843,094 |
| 3 | 128 | 86 | 12 | 206.4 | 1,699,542 |
| 4 | 2 | 86 | 15 | 123.1 | 36,852 |
| 4 | 4 | 86 | 72 | 473.1 | 62,514 |
| 4 | 8 | 86 | 14 | 127.3 | 113,934 |
| 4 | 16 | 86 | 13 | 21.8 | 217,158 |
| 4 | 32 | 86 | 13 | 46.4 | 425,142 |
| 4 | 64 | 86 | 12 | 77.0 | 847,254 |
| 4 | 128 | 86 | 13 | 210.5 | 1,716,054 |
| 5 | 2 | 86 | 72 | 481.6 | 36,858 |
| 5 | 4 | 86 | 23 | 178.1 | 62,534 |
| 5 | 8 | 86 | 18 | 158.9 | 114,006 |
| 5 | 16 | 86 | 11 | 21.3 | 217,430 |
| 5 | 32 | 86 | 13 | 51.7 | 426,198 |
| 5 | 64 | 86 | 13 | 99.0 | 851,414 |
| 5 | 128 | 86 | 11 | 172.7 | 1,732,566 |
| 6 | 2 | 86 | 71 | 532.3 | 36,864 |
| 6 | 4 | 86 | 18 | 177.5 | 62,554 |
| 6 | 8 | 86 | 22 | 247.4 | 114,078 |
| 6 | 16 | 86 | 16 | 27.8 | 217,702 |
| 6 | 32 | 86 | 16 | 58.1 | 427,254 |
| 6 | 64 | 86 | 13 | 96.8 | 855,574 |
| 6 | 128 | 86 | 24 | 459.0 | 1,749,078 |

*Table C.26: Training Metrics for N = 10 (Indistinguishable).*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.096 | 0.058 | 0.191 | 0.443 | 0.199 |
| 1 | 4 | 8.044 | 0.069 | 0.209 | 0.463 | 0.240 |

*Continued on next page*

*Table C.26 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 8 | 7.475 | 0.153 | 0.298 | 0.518 | 0.403 |
| 1 | 16 | 7.328 | 0.180 | 0.328 | 0.535 | 0.447 |
| 1 | 32 | 7.060 | 0.237 | 0.385 | 0.564 | 0.539 |
| 1 | 64 | 6.496 | 0.330 | 0.503 | 0.652 | 0.660 |
| 1 | 128 | 5.796 | 0.490 | 0.656 | 0.771 | 0.808 |
| 2 | 2 | 8.248 | 0.024 | 0.163 | 0.461 | 0.122 |
| 2 | 4 | 7.831 | 0.095 | 0.236 | 0.467 | 0.299 |
| 2 | 8 | 7.764 | 0.112 | 0.250 | 0.477 | 0.329 |
| 2 | 16 | 7.095 | 0.226 | 0.386 | 0.568 | 0.524 |
| 2 | 32 | 6.357 | 0.378 | 0.540 | 0.670 | 0.707 |
| 2 | 64 | 6.219 | 0.379 | 0.551 | 0.688 | 0.712 |
| 2 | 128 | 5.416 | 0.492 | 0.706 | 0.806 | 0.826 |
| 3 | 2 | 8.210 | 0.039 | 0.182 | 0.458 | 0.154 |
| 3 | 4 | 8.209 | 0.034 | 0.169 | 0.451 | 0.165 |
| 3 | 8 | 7.505 | 0.148 | 0.293 | 0.493 | 0.405 |
| 3 | 16 | 7.512 | 0.141 | 0.306 | 0.517 | 0.387 |
| 3 | 32 | 5.907 | 0.444 | 0.625 | 0.746 | 0.765 |
| 3 | 64 | 5.567 | 0.523 | 0.706 | 0.797 | 0.832 |
| 3 | 128 | 4.259 | 0.745 | 0.921 | 0.948 | 0.967 |
| 4 | 2 | 8.233 | 0.035 | 0.171 | 0.457 | 0.141 |
| 4 | 4 | 8.248 | 0.024 | 0.157 | 0.450 | 0.122 |
| 4 | 8 | 7.745 | 0.111 | 0.257 | 0.480 | 0.325 |
| 4 | 16 | 7.183 | 0.182 | 0.358 | 0.553 | 0.477 |
| 4 | 32 | 5.998 | 0.407 | 0.610 | 0.719 | 0.749 |
| 4 | 64 | 5.174 | 0.577 | 0.765 | 0.835 | 0.880 |
| 4 | 128 | 4.206 | 0.783 | 0.931 | 0.953 | 0.975 |
| 5 | 2 | 8.248 | 0.024 | 0.163 | 0.461 | 0.122 |
| 5 | 4 | 7.896 | 0.086 | 0.232 | 0.465 | 0.276 |
| 5 | 8 | 7.812 | 0.093 | 0.238 | 0.479 | 0.294 |
| 5 | 16 | 7.941 | 0.076 | 0.226 | 0.468 | 0.253 |
| 5 | 32 | 6.803 | 0.254 | 0.428 | 0.590 | 0.563 |
| 5 | 64 | 5.729 | 0.484 | 0.670 | 0.766 | 0.807 |
| 5 | 128 | 6.243 | 0.364 | 0.557 | 0.681 | 0.700 |
| 6 | 2 | 8.248 | 0.024 | 0.163 | 0.461 | 0.122 |
| 6 | 4 | 8.248 | 0.024 | 0.157 | 0.450 | 0.122 |
| 6 | 8 | 7.666 | 0.105 | 0.267 | 0.500 | 0.330 |

*Continued on next page*

*Table C.26 – continued from previous page*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 6 | 16 | 7.531 | 0.139 | 0.301 | 0.521 | 0.381 |
| 6 | 32 | 7.224 | 0.167 | 0.348 | 0.542 | 0.442 |
| 6 | 64 | 7.089 | 0.189 | 0.365 | 0.572 | 0.478 |
| 6 | 128 | 6.771 | 0.209 | 0.422 | 0.617 | 0.512 |

*Table C.27: Validation Metrics for N = 10 (Indistinguishable).*

| Layers | Filters | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 1 | 2 | 8.322 | 0.018 | 0.150 | 0.437 | 0.112 |
| 1 | 4 | 8.331 | 0.022 | 0.156 | 0.431 | 0.105 |
| 1 | 8 | 8.635 | 0.017 | 0.161 | 0.420 | 0.106 |
| 1 | 16 | 8.725 | 0.025 | 0.153 | 0.402 | 0.105 |
| 1 | 32 | 8.952 | 0.021 | 0.144 | 0.401 | 0.105 |
| 1 | 64 | 9.224 | 0.028 | 0.152 | 0.403 | 0.109 |
| 1 | 128 | 9.551 | 0.022 | 0.144 | 0.388 | 0.101 |
| 2 | 2 | 8.249 | 0.021 | 0.171 | 0.459 | 0.121 |
| 2 | 4 | 8.498 | 0.021 | 0.151 | 0.424 | 0.094 |
| 2 | 8 | 8.523 | 0.018 | 0.148 | 0.427 | 0.105 |
| 2 | 16 | 8.914 | 0.018 | 0.141 | 0.390 | 0.094 |
| 2 | 32 | 9.435 | 0.022 | 0.137 | 0.386 | 0.105 |
| 2 | 64 | 9.270 | 0.020 | 0.159 | 0.411 | 0.105 |
| 2 | 128 | 9.923 | 0.024 | 0.152 | 0.417 | 0.112 |
| 3 | 2 | 8.252 | 0.021 | 0.162 | 0.456 | 0.122 |
| 3 | 4 | 8.263 | 0.024 | 0.158 | 0.453 | 0.113 |
| 3 | 8 | 8.733 | 0.017 | 0.147 | 0.405 | 0.093 |
| 3 | 16 | 8.530 | 0.026 | 0.151 | 0.429 | 0.120 |
| 3 | 32 | 9.406 | 0.027 | 0.152 | 0.428 | 0.116 |
| 3 | 64 | 9.781 | 0.023 | 0.147 | 0.434 | 0.110 |
| 3 | 128 | 11.270 | 0.026 | 0.148 | 0.429 | 0.115 |
| 4 | 2 | 8.250 | 0.018 | 0.167 | 0.459 | 0.124 |
| 4 | 4 | 8.249 | 0.024 | 0.157 | 0.460 | 0.121 |
| 4 | 8 | 8.553 | 0.015 | 0.141 | 0.424 | 0.102 |
| 4 | 16 | 8.770 | 0.022 | 0.138 | 0.414 | 0.113 |
| 4 | 32 | 9.620 | 0.020 | 0.128 | 0.365 | 0.094 |
| 4 | 64 | 10.376 | 0.018 | 0.128 | 0.392 | 0.094 |

*Continued on next page*

*Table C.27 – continued from previous page*

| Layers | Filters | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|
| 4 | 128 | 11.413 | 0.019 | 0.132 | 0.409 | 0.102 |
| 5 | 2 | 8.249 | 0.021 | 0.171 | 0.459 | 0.121 |
| 5 | 4 | 8.470 | 0.024 | 0.147 | 0.406 | 0.106 |
| 5 | 8 | 8.439 | 0.022 | 0.161 | 0.454 | 0.119 |
| 5 | 16 | 8.352 | 0.019 | 0.150 | 0.423 | 0.107 |
| 5 | 32 | 8.996 | 0.022 | 0.143 | 0.408 | 0.106 |
| 5 | 64 | 9.519 | 0.018 | 0.135 | 0.411 | 0.107 |
| 5 | 128 | 9.312 | 0.020 | 0.143 | 0.407 | 0.107 |
| 6 | 2 | 8.249 | 0.021 | 0.171 | 0.459 | 0.121 |
| 6 | 4 | 8.249 | 0.024 | 0.157 | 0.460 | 0.121 |
| 6 | 8 | 8.393 | 0.016 | 0.158 | 0.451 | 0.120 |
| 6 | 16 | 8.601 | 0.022 | 0.136 | 0.410 | 0.103 |
| 6 | 32 | 8.737 | 0.024 | 0.144 | 0.412 | 0.116 |
| 6 | 64 | 8.703 | 0.019 | 0.145 | 0.436 | 0.111 |
| 6 | 128 | 8.878 | 0.018 | 0.157 | 0.438 | 0.117 |

# Appendix D

# Network Results for Partially Indistinguishable Permanents

*Table D.1: Model Metrics for Partially Indistinguishable Permanents.*

| N | L | C | Index | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|---|---|
| 3 | 3 | 8 | 0.0 | 57 | 269 | 102.5 | 11,865 |
| 3 | 3 | 8 | 0.1 | 57 | 148 | 58.6 | 11,865 |
| 3 | 3 | 8 | 0.2 | 57 | 176 | 75.6 | 11,865 |
| 3 | 3 | 8 | 0.3 | 57 | 165 | 83.8 | 11,865 |
| 3 | 3 | 8 | 0.4 | 57 | 149 | 62.3 | 11,865 |
| 3 | 3 | 8 | 0.5 | 57 | 167 | 66.8 | 11,865 |
| 3 | 3 | 8 | 0.6 | 57 | 189 | 76.7 | 11,865 |
| 3 | 3 | 8 | 0.7 | 58 | 148 | 62.3 | 11,994 |
| 3 | 3 | 8 | 0.8 | 60 | 102 | 44.3 | 12,252 |
| 3 | 3 | 8 | 0.9 | 64 | 132 | 61.1 | 12,768 |
| 3 | 3 | 8 | 1.0 | 76 | 65 | 33.9 | 14,316 |
| 4 | 4 | 16 | 0.0 | 53 | 130 | 63.6 | 26,629 |
| 4 | 4 | 16 | 0.1 | 53 | 150 | 75.4 | 26,629 |
| 4 | 4 | 16 | 0.2 | 53 | 191 | 91.2 | 26,629 |
| 4 | 4 | 16 | 0.3 | 53 | 121 | 62.0 | 26,629 |
| 4 | 4 | 16 | 0.4 | 54 | 95 | 49.6 | 26,758 |
| 4 | 4 | 16 | 0.5 | 54 | 114 | 57.5 | 26,758 |
| 4 | 4 | 16 | 0.6 | 54 | 103 | 53.6 | 26,758 |
| 4 | 4 | 16 | 0.7 | 56 | 55 | 30.0 | 27,016 |
| 4 | 4 | 16 | 0.8 | 58 | 75 | 38.5 | 27,274 |
| 4 | 4 | 16 | 0.9 | 62 | 32 | 18.3 | 27,790 |
| 4 | 4 | 16 | 1.0 | 83 | 38 | 25.0 | 30,499 |

*Continued on next page*

*Table D.1 – continued from previous page*

| N | L | C | Index | Classes | Epochs | Training Time (s) | Parameters |
|---|---|---|---|---|---|---|---|
| 5 | 5 | 64 | 0.0 | 53 | 88 | 147.4 | 159,413 |
| 5 | 5 | 64 | 0.1 | 53 | 89 | 142.2 | 159,413 |
| 5 | 5 | 64 | 0.2 | 53 | 73 | 116.2 | 159,413 |
| 5 | 5 | 64 | 0.3 | 53 | 53 | 94.5 | 159,413 |
| 5 | 5 | 64 | 0.4 | 53 | 80 | 129.7 | 159,413 |
| 5 | 5 | 64 | 0.5 | 54 | 78 | 131.5 | 159,542 |
| 5 | 5 | 64 | 0.6 | 54 | 61 | 102.6 | 159,542 |
| 5 | 5 | 64 | 0.7 | 55 | 28 | 49.4 | 159,671 |
| 5 | 5 | 64 | 0.8 | 57 | 44 | 71.7 | 159,929 |
| 5 | 5 | 64 | 0.9 | 61 | 27 | 44.0 | 160,445 |
| 5 | 5 | 64 | 1.0 | 83 | 27 | 46.7 | 163,283 |
| 6 | 6 | 64 | 0.0 | 52 | 49 | 111.0 | 237,172 |
| 6 | 6 | 64 | 0.1 | 52 | 24 | 54.3 | 237,172 |
| 6 | 6 | 64 | 0.2 | 52 | 47 | 108.8 | 237,172 |
| 6 | 6 | 64 | 0.3 | 52 | 34 | 82.5 | 237,172 |
| 6 | 6 | 64 | 0.4 | 52 | 39 | 90.2 | 237,172 |
| 6 | 6 | 64 | 0.5 | 52 | 29 | 69.6 | 237,172 |
| 6 | 6 | 64 | 0.6 | 54 | 33 | 78.5 | 237,430 |
| 6 | 6 | 64 | 0.7 | 54 | 31 | 75.9 | 237,430 |
| 6 | 6 | 64 | 0.8 | 57 | 26 | 67.9 | 237,817 |
| 6 | 6 | 64 | 0.9 | 61 | 29 | 69.0 | 238,333 |
| 6 | 6 | 64 | 1.0 | 82 | 24 | 57.7 | 241,042 |
| 7 | 5 | 16 | 0.0 | 53 | 29 | 29.0 | 82,197 |
| 7 | 5 | 16 | 0.1 | 53 | 44 | 38.9 | 82,197 |
| 7 | 5 | 16 | 0.2 | 53 | 50 | 49.4 | 82,197 |
| 7 | 5 | 16 | 0.3 | 53 | 76 | 69.2 | 82,197 |
| 7 | 5 | 16 | 0.4 | 53 | 42 | 42.1 | 82,197 |
| 7 | 5 | 16 | 0.5 | 53 | 57 | 53.2 | 82,197 |
| 7 | 5 | 16 | 0.6 | 54 | 41 | 40.3 | 82,326 |
| 7 | 5 | 16 | 0.7 | 55 | 35 | 33.2 | 82,455 |
| 7 | 5 | 16 | 0.8 | 57 | 37 | 35.1 | 82,713 |
| 7 | 5 | 16 | 0.9 | 61 | 36 | 35.8 | 83,229 |
| 7 | 5 | 16 | 1.0 | 85 | 26 | 35.2 | 86,325 |

*Table D.2: Training Metrics for Partially Indistinguishable Permanents.*

| N | L | C | Index | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 8 | 0.0 | 5.754 | 0.142 | 0.541 | 0.939 | 0.484 |
| 3 | 3 | 8 | 0.1 | 6.189 | 0.103 | 0.461 | 0.894 | 0.392 |
| 3 | 3 | 8 | 0.2 | 5.899 | 0.136 | 0.522 | 0.930 | 0.467 |
| 3 | 3 | 8 | 0.3 | 6.173 | 0.108 | 0.462 | 0.895 | 0.397 |
| 3 | 3 | 8 | 0.4 | 5.993 | 0.123 | 0.488 | 0.922 | 0.429 |
| 3 | 3 | 8 | 0.5 | 6.174 | 0.108 | 0.455 | 0.888 | 0.395 |
| 3 | 3 | 8 | 0.6 | 6.146 | 0.114 | 0.456 | 0.899 | 0.404 |
| 3 | 3 | 8 | 0.7 | 6.454 | 0.093 | 0.391 | 0.843 | 0.341 |
| 3 | 3 | 8 | 0.8 | 6.809 | 0.082 | 0.326 | 0.745 | 0.300 |
| 3 | 3 | 8 | 0.9 | 6.898 | 0.076 | 0.300 | 0.723 | 0.292 |
| 3 | 3 | 8 | 1.0 | 7.447 | 0.070 | 0.263 | 0.614 | 0.237 |
| 4 | 4 | 16 | 0.0 | 5.142 | 0.187 | 0.689 | 0.985 | 0.621 |
| 4 | 4 | 16 | 0.1 | 5.161 | 0.190 | 0.690 | 0.987 | 0.614 |
| 4 | 4 | 16 | 0.2 | 5.103 | 0.196 | 0.704 | 0.989 | 0.641 |
| 4 | 4 | 16 | 0.3 | 5.289 | 0.163 | 0.659 | 0.986 | 0.582 |
| 4 | 4 | 16 | 0.4 | 5.331 | 0.167 | 0.644 | 0.982 | 0.562 |
| 4 | 4 | 16 | 0.5 | 5.396 | 0.153 | 0.626 | 0.979 | 0.558 |
| 4 | 4 | 16 | 0.6 | 5.544 | 0.150 | 0.594 | 0.969 | 0.529 |
| 4 | 4 | 16 | 0.7 | 5.999 | 0.111 | 0.493 | 0.930 | 0.414 |
| 4 | 4 | 16 | 0.8 | 6.256 | 0.111 | 0.412 | 0.865 | 0.388 |
| 4 | 4 | 16 | 0.9 | 6.741 | 0.076 | 0.318 | 0.770 | 0.284 |
| 4 | 4 | 16 | 1.0 | 7.512 | 0.065 | 0.256 | 0.603 | 0.242 |
| 5 | 5 | 64 | 0.0 | 4.540 | 0.283 | 0.813 | 0.996 | 0.775 |
| 5 | 5 | 64 | 0.1 | 4.560 | 0.274 | 0.811 | 0.996 | 0.773 |
| 5 | 5 | 64 | 0.2 | 4.756 | 0.243 | 0.777 | 0.996 | 0.718 |
| 5 | 5 | 64 | 0.3 | 5.300 | 0.177 | 0.633 | 0.985 | 0.582 |
| 5 | 5 | 64 | 0.4 | 4.889 | 0.232 | 0.742 | 0.992 | 0.694 |
| 5 | 5 | 64 | 0.5 | 4.932 | 0.228 | 0.725 | 0.992 | 0.683 |
| 5 | 5 | 64 | 0.6 | 5.319 | 0.183 | 0.627 | 0.982 | 0.583 |
| 5 | 5 | 64 | 0.7 | 6.088 | 0.114 | 0.458 | 0.909 | 0.407 |
| 5 | 5 | 64 | 0.8 | 6.163 | 0.118 | 0.448 | 0.884 | 0.413 |
| 5 | 5 | 64 | 0.9 | 6.628 | 0.100 | 0.362 | 0.784 | 0.341 |
| 5 | 5 | 64 | 1.0 | 7.502 | 0.074 | 0.262 | 0.582 | 0.265 |
| 6 | 6 | 64 | 0.0 | 5.232 | 0.175 | 0.650 | 0.992 | 0.572 |
| 6 | 6 | 64 | 0.1 | 5.260 | 0.180 | 0.668 | 0.983 | 0.598 |

*Continued on next page*

*Table D.2 – continued from previous page*

| N | L | C | Index | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|---|---|
| 6 | 6 | 64 | 0.2 | 4.902 | 0.224 | 0.733 | 0.993 | 0.695 |
| 6 | 6 | 64 | 0.3 | 5.444 | 0.157 | 0.594 | 0.986 | 0.534 |
| 6 | 6 | 64 | 0.4 | 5.206 | 0.186 | 0.669 | 0.986 | 0.614 |
| 6 | 6 | 64 | 0.5 | 5.469 | 0.162 | 0.592 | 0.969 | 0.543 |
| 6 | 6 | 64 | 0.6 | 5.520 | 0.159 | 0.579 | 0.963 | 0.533 |
| 6 | 6 | 64 | 0.7 | 5.856 | 0.139 | 0.522 | 0.933 | 0.471 |
| 6 | 6 | 64 | 0.8 | 6.119 | 0.121 | 0.463 | 0.899 | 0.423 |
| 6 | 6 | 64 | 0.9 | 6.524 | 0.105 | 0.376 | 0.796 | 0.367 |
| 6 | 6 | 64 | 1.0 | 7.556 | 0.080 | 0.249 | 0.557 | 0.268 |
| 7 | 5 | 16 | 0.0 | 5.533 | 0.148 | 0.590 | 0.974 | 0.522 |
| 7 | 5 | 16 | 0.1 | 5.668 | 0.136 | 0.535 | 0.971 | 0.477 |
| 7 | 5 | 16 | 0.2 | 5.504 | 0.163 | 0.605 | 0.973 | 0.538 |
| 7 | 5 | 16 | 0.3 | 5.293 | 0.201 | 0.642 | 0.978 | 0.604 |
| 7 | 5 | 16 | 0.4 | 5.488 | 0.175 | 0.598 | 0.972 | 0.552 |
| 7 | 5 | 16 | 0.5 | 5.513 | 0.177 | 0.598 | 0.968 | 0.558 |
| 7 | 5 | 16 | 0.6 | 5.681 | 0.154 | 0.561 | 0.951 | 0.510 |
| 7 | 5 | 16 | 0.7 | 6.038 | 0.145 | 0.483 | 0.897 | 0.460 |
| 7 | 5 | 16 | 0.8 | 6.180 | 0.140 | 0.446 | 0.868 | 0.443 |
| 7 | 5 | 16 | 0.9 | 6.726 | 0.098 | 0.340 | 0.738 | 0.342 |
| 7 | 5 | 16 | 1.0 | 7.448 | 0.110 | 0.287 | 0.555 | 0.337 |

*Table D.3: Validation Metrics for Partially Indistinguishable Permanents.*

| N | L | C | Index | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 8 | 0.0 | 5.959 | 0.102 | 0.513 | 0.926 | 0.408 |
| 3 | 3 | 8 | 0.1 | 6.253 | 0.071 | 0.436 | 0.901 | 0.337 |
| 3 | 3 | 8 | 0.2 | 6.255 | 0.078 | 0.467 | 0.909 | 0.347 |
| 3 | 3 | 8 | 0.3 | 6.362 | 0.072 | 0.429 | 0.881 | 0.323 |
| 3 | 3 | 8 | 0.4 | 6.234 | 0.085 | 0.473 | 0.899 | 0.357 |
| 3 | 3 | 8 | 0.5 | 6.404 | 0.066 | 0.411 | 0.865 | 0.309 |
| 3 | 3 | 8 | 0.6 | 6.395 | 0.061 | 0.403 | 0.878 | 0.303 |
| 3 | 3 | 8 | 0.7 | 6.563 | 0.068 | 0.372 | 0.848 | 0.290 |
| 3 | 3 | 8 | 0.8 | 6.901 | 0.053 | 0.313 | 0.744 | 0.233 |
| 3 | 3 | 8 | 0.9 | 7.086 | 0.055 | 0.269 | 0.713 | 0.216 |
| 3 | 3 | 8 | 1.0 | 7.615 | 0.042 | 0.229 | 0.609 | 0.159 |

*Continued on next page*

*Table D.3 – continued from previous page*

| N | L | C | Index | Loss | Acc (±0) | Acc (±3) | Acc (±10) | Acc (Top-5) |
|---|---|---|---|---|---|---|---|---|
| 4 | 4 | 16 | 0.0 | 5.454 | 0.115 | 0.640 | 0.982 | 0.504 |
| 4 | 4 | 16 | 0.1 | 5.424 | 0.121 | 0.636 | 0.981 | 0.514 |
| 4 | 4 | 16 | 0.2 | 5.353 | 0.126 | 0.651 | 0.986 | 0.529 |
| 4 | 4 | 16 | 0.3 | 5.552 | 0.113 | 0.610 | 0.979 | 0.472 |
| 4 | 4 | 16 | 0.4 | 5.593 | 0.102 | 0.601 | 0.970 | 0.480 |
| 4 | 4 | 16 | 0.5 | 5.760 | 0.088 | 0.551 | 0.964 | 0.425 |
| 4 | 4 | 16 | 0.6 | 5.800 | 0.098 | 0.549 | 0.957 | 0.423 |
| 4 | 4 | 16 | 0.7 | 6.180 | 0.070 | 0.446 | 0.921 | 0.341 |
| 4 | 4 | 16 | 0.8 | 6.553 | 0.061 | 0.344 | 0.842 | 0.268 |
| 4 | 4 | 16 | 0.9 | 6.915 | 0.045 | 0.293 | 0.748 | 0.217 |
| 4 | 4 | 16 | 1.0 | 7.696 | 0.034 | 0.228 | 0.578 | 0.158 |
| 5 | 5 | 64 | 0.0 | 5.514 | 0.134 | 0.665 | 0.985 | 0.543 |
| 5 | 5 | 64 | 0.1 | 5.484 | 0.118 | 0.657 | 0.990 | 0.531 |
| 5 | 5 | 64 | 0.2 | 5.605 | 0.112 | 0.643 | 0.987 | 0.494 |
| 5 | 5 | 64 | 0.3 | 5.916 | 0.092 | 0.532 | 0.960 | 0.408 |
| 5 | 5 | 64 | 0.4 | 5.645 | 0.119 | 0.625 | 0.984 | 0.492 |
| 5 | 5 | 64 | 0.5 | 5.727 | 0.100 | 0.583 | 0.972 | 0.438 |
| 5 | 5 | 64 | 0.6 | 5.994 | 0.088 | 0.518 | 0.950 | 0.406 |
| 5 | 5 | 64 | 0.7 | 6.271 | 0.071 | 0.419 | 0.890 | 0.332 |
| 5 | 5 | 64 | 0.8 | 6.437 | 0.066 | 0.377 | 0.869 | 0.290 |
| 5 | 5 | 64 | 0.9 | 7.013 | 0.043 | 0.298 | 0.742 | 0.218 |
| 5 | 5 | 64 | 1.0 | 7.889 | 0.030 | 0.215 | 0.548 | 0.144 |
| 6 | 6 | 64 | 0.0 | 5.813 | 0.097 | 0.572 | 0.983 | 0.427 |
| 6 | 6 | 64 | 0.1 | 5.541 | 0.121 | 0.613 | 0.978 | 0.495 |
| 6 | 6 | 64 | 0.2 | 5.500 | 0.111 | 0.647 | 0.984 | 0.500 |
| 6 | 6 | 64 | 0.3 | 5.890 | 0.100 | 0.537 | 0.975 | 0.397 |
| 6 | 6 | 64 | 0.4 | 5.590 | 0.093 | 0.588 | 0.978 | 0.447 |
| 6 | 6 | 64 | 0.5 | 5.806 | 0.099 | 0.551 | 0.960 | 0.429 |
| 6 | 6 | 64 | 0.6 | 5.892 | 0.083 | 0.515 | 0.951 | 0.394 |
| 6 | 6 | 64 | 0.7 | 6.118 | 0.086 | 0.450 | 0.922 | 0.360 |
| 6 | 6 | 64 | 0.8 | 6.463 | 0.066 | 0.397 | 0.870 | 0.287 |
| 6 | 6 | 64 | 0.9 | 6.857 | 0.043 | 0.298 | 0.767 | 0.220 |
| 6 | 6 | 64 | 1.0 | 7.988 | 0.032 | 0.177 | 0.508 | 0.129 |
| 7 | 5 | 16 | 0.0 | 5.681 | 0.112 | 0.580 | 0.969 | 0.454 |
| 7 | 5 | 16 | 0.1 | 5.798 | 0.088 | 0.527 | 0.972 | 0.404 |
| 7 | 5 | 16 | 0.2 | 5.730 | 0.095 | 0.566 | 0.962 | 0.406 |

*Continued on next page*

*Table D.3 – continued from previous page*

| N | L | C | Index | Loss | Acc ($\pm$0) | Acc ($\pm$3) | Acc ($\pm$10) | Acc (Top-5) |
|---|---|---|---|---|---|---|---|---|
| 7 | 5 | 16 | 0.3 | 5.719 | 0.103 | 0.548 | 0.977 | 0.423 |
| 7 | 5 | 16 | 0.4 | 5.777 | 0.080 | 0.527 | 0.967 | 0.398 |
| 7 | 5 | 16 | 0.5 | 5.839 | 0.089 | 0.517 | 0.961 | 0.400 |
| 7 | 5 | 16 | 0.6 | 6.041 | 0.090 | 0.485 | 0.933 | 0.364 |
| 7 | 5 | 16 | 0.7 | 6.357 | 0.070 | 0.417 | 0.882 | 0.318 |
| 7 | 5 | 16 | 0.8 | 6.532 | 0.062 | 0.386 | 0.848 | 0.287 |
| 7 | 5 | 16 | 0.9 | 6.988 | 0.055 | 0.292 | 0.720 | 0.230 |
| 7 | 5 | 16 | 1.0 | 8.206 | 0.026 | 0.171 | 0.475 | 0.121 |