Using Curriculum learning to improve the performance of Deep Learning Models used for Classification Purposes

Univeristy of Twente

R.S.J.Molenaar

UNIVERSITY
OF TWENTE.

1 Abstract

Keywords: *Curriculum learning, Age of Aquisition, Deep Learning, Convolutional Network, ResNet50, ImageNet*

The current growth of neural networks means that the importance of the speed and accuracy of these neural networks, especially during training, is becoming more and more important. Although the construction of Convolutional Neural Networks has grown a lot, with pruning nodes, removing completely linked layers, and cutting down on filters, the training processes of these neural networks leaves a lot to be desired. This paper focuses on increasing the training performance and speed of neural networks using a technique called curriculum learning. This method was formulated to represent the manner in which humans learn, starting with easier concepts and following it up with harder ones. To use this strategy, a curriculum must be made based on a metric. In this research, this metric was chosen to be the Age of Acquisition for words, ranking concepts on at what age humans learn words. Both easy and hard AoA classes were tested on accuracy and training performance, together with multiple test as a baseline. The results show a significant improvement, but further research must be done to confirm it and to explore this idea further.

Contents

2 Introduction

2.1 Deep Learning

Deep learning, Artificial Intelligence and Neural Networks are more and more prevalent in the ever changing digital industry. Deep learning constitutes a modern technique for image processing and data analysis, with a large potential. An important advantage of using Deep Learning in image processing is the reduced need of feature engineering [1, 20], because the important features are located automatically by the algorithm through training, instead of manually implemented. Without a doubt, the amount of applications possible for deep learning are boundless, however, the learning process requires large training sets and considerable resources in computation, energy consumption and time.

Because of the wide range of applicable scenarios of Deep Learning, the use of and interest in this technique is increasing rapidly [2], therefore increasing the performance of these networks is an important challenge. Hence, this paper will focus on providing an overview of research done on the improvements of the performance of Deep Learning models, especially convolutional neural networks, with regards to the difference in training methods available. Focusing on how curriculum learning, e.g. first learning basic concepts before learning more difficult objectives, can affect this process. The literary review will provide an understanding of curriculum learning and to give an overview of different methods of training a Neural Network with curriculum learning. The first part will serve as an explanation of curriculum learning. Following this, this review will focus on the effect curriculum learning can have on a convolutional neural network. From there, an experiment using curriculum learning based on a concept called Age of Acquisition will be conducted.

2.2 Curriculum Learning

Curriculum learning is a tool that is being used more and more in the field of Deep Learning and Neural Networks. Bengio et al.[7] describes curriculum learning as a "starting small" strategy. This strategy is reflected in schools all around the world as teachers start with typical 'easy' examples first, and later go on to explain more ambiguous examples [16]. In order to execute this task, teachers often create a curriculum. The process of Curriculum Learning thus, means learning through a meaningful order of concepts or examples, instead of randomly picking one concept and learning those before moving on. We as humans have shown to greatly improve our learning speed this way.

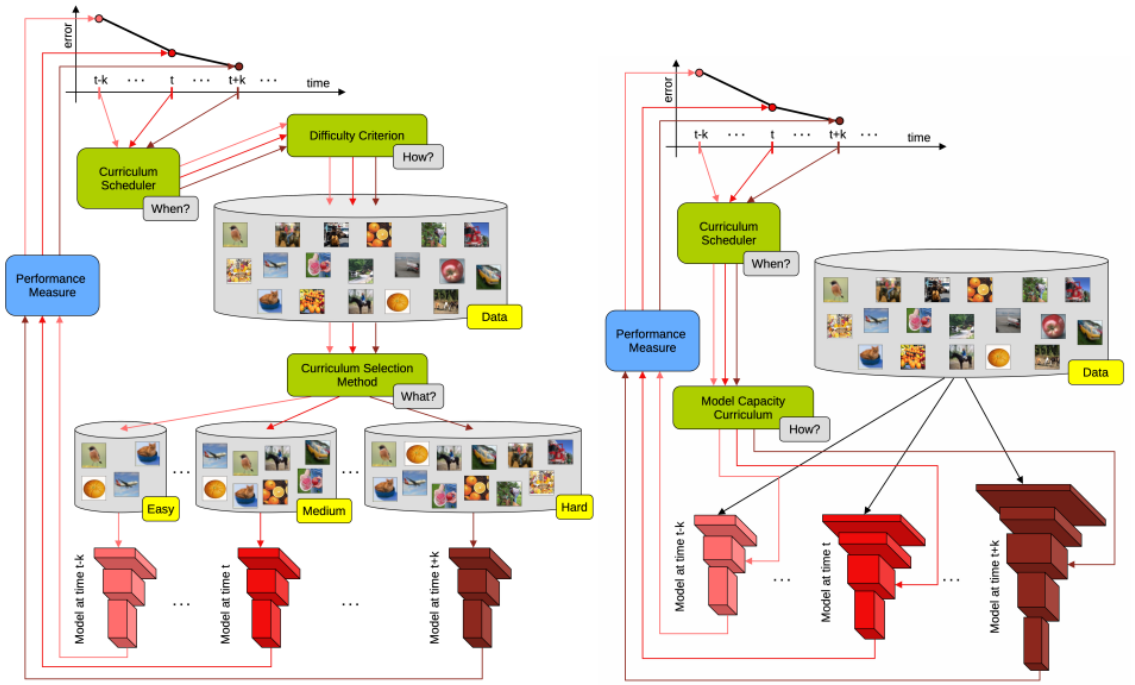
3 Background

3.1 History

The idea to use this strategy as a tool for training machine learning algorithms in the same way can at least be resolved back to Elman in 1993 [17]. They concluded that starting with an architecture which was restrained in its complexity, and gradually increasing that complexity, could lead to more successful results. Related ideas have been explored a lot in the years since then. The first to formally establish this gradual increase in difficulty into Deep Learning was Bengio et al [7]. They evaluated two experiments, one on geometric shapes, the other on language modelling. The geometric shape networks was asked to classify 3 different classes of shapes: rectangles, ellipses, and triangles. The curriculum was implemented by dividing the data set in two, a *BasicShapes* set containing only squares, circles and equilateral triangles, and a *GeomShapes* set containing rectangles, ellipses and all kinds of triangles. Furthermore, the second set showed not only more variety in shape, but also differences in position, size, orientation and were blended more into the background. Then, by first giving the network the set of *Basicshapes* and afterwards inputting the *GeomShapes* set gave significantly better results than presenting the algorithm with a mixture of the two sets at once, right from the beginning. The second experiment involved predicting the best word which could follow a given context of word. Here the curriculum was implemented by slowly increasing the data set of words to give a score to, based on how well they would fit. Meaning the curriculum version steadily grew the vocabulary by 5.000 words after each pass on the context. In this experiment, they also observed an improvement in accuracy with this curriculum trained model. Since this paper, curriculum learning has been used in more than 150 academic scenarios involving machine learning, from object detection, neural machine translation and speech recognition.

3.2 Curriculum Learning Methods

While there has always been a clear consensus of the definition of curriculum learning in the context of Neural Networks, the approach to implementing this learning strategy varies. Soviany et al [8] describes the four main components of any deep learning algorithm as: *'the data, the model, the task and the performance measure'*. They also stated that curriculum learning can be applied to each of these components. Avramova (2015), Hacohen et al (2019), Zhang et al (2018) and Wang (2019) [4-6,12] all think of curriculum learning as a tool to structure the data, or more specifically the order of the data. This approach is known as the 'natural approach' to curriculum learning [8], which applies curriculum learning on the data level, and involves steadily giving the neural network more difficult concepts during the training process (Figure 1.a). Another method proposed by Karras et al. [13] is to grow the capacity of the model by adding or activating more neural units during the training process to accelerate the models ability to interpret data (Figure 1.b), which is applying curriculum learning to the model level.



a General framework for data-level curriculum learning.

b General framework for model-level curriculum.

Figure 1: Two different approaches to curriculum learning, one on the data-level, and the other (b) on the model-level [8]

However, each method does have it's positives and negatives. An issue with the model-level approach to curriculum learning that the natural approach does not suffer from, is that the increase in the capacity of the model is not dependent on the difficulty of the training data, but only on the size, which means it is an easier general approach, but for optimal performance, the natural approach can be more suited. A crucial part of finding a suitable approach is creating the right curriculum, so either creating an accurate classification teacher model, or to accurately classifying the data from easy to difficult. This aspect is essential for improvements in performance [6]. The natural approach, for instance, suffers from the fact that the curriculum selection has to be implemented, either externally, or through an internal process. The problem with having to implement this curriculum externally is that external qualification is not always possible. But this disadvantage of the natural approach can also be solved in a number of ways. The problem of qualification of difficulty is a big topic related to, and involved in, curriculum learning, and multiple means have been implemented to help this aspect of curriculum learning. Currently there have been five most used 'categories' of Curriculum Learning in literature:

3.2.1 Vanilla Curriculum Learning

Vanilla Curriculum Learning is mostly used when a data set has clearly defined labels of difficulty beforehand. This strategy only has an influence on the data level of the model, structuring it in an ascending manner, based on this label of difficulty. An example of this is Bengio et al. [7], where

they used this approach in their geometric shapes qualifier to split the data into two sets, one containing easier and clearer images of 'perfect' shapes, and the other containing more convoluted shapes and sizes. And using the easy dataset to gain an advantage on the accuracy of the model when tested on the second set.

3.2.2 Self-Paced Learning

Self-paced learning is a more dynamic and advanced method for implementing Curriculum Learning in Neural Networks. This strategy is dynamic in the sense that during training, the difficulty is measured and with this measure, the order of the training samples is altered to further improve the training process. This difficulty can be measured in a number of ways, with one of the more common values being the loss function of samples. For instance, Avramova [4] looked into using Self Paced Learning strategies to determine difficulty of a sample using the sample's loss function calculated based on the goal of the network. They found that all different versions of SPL performed within 1 per cent of each other. But most importantly that inverse SPL networks performed slightly better, and had '*marginally higher accuracy results*'

3.2.3 Self-Paced Curriculum Learning

Self-Paced Curriculum is a combination of the first two, combining both initial labels to define the order of samples, but also dynamically changing this order using difficulty measures during training. This combination was first implemented by Jaing et al. [21] where they found that Self-Paced Learning strategy missed 'a way to incorporate prior guidance in learning'[21].

3.2.4 Progressive Curriculum Learning

Progressive Curriculum Learning differs from the previous methods in the way that those all changed the order of the data during training, and individually calculated this difficulty per sample. Progressive Curriculum Learning is a strategy where this individual difficulty is not used as a measure for the network, but rather, it uses the collective difficulty of the task setting to gradually increase or decrease settings in the algorithm, either in the model or in the task to create an improvement in training efficiency. Karras et al. [13] uses this more progressive form of CL to it's advantage in gradually growing the scope of a GAN (Generative Adversarial Network) to obtain higher quality results.

3.2.5 Teacher-Student Curriculum Learning

Both Hacohen et al. [5] and Kim et al. [14] use a type of curriculum learning which is usually classified as teacher-student CL, which divides the training process of the model into two separate tasks. One of those will still focus on the primary task (student) while the teacher subsection will focus on defining the optimal hyperparameters and training process for the student. The first to propose this strategy was Kim et al. [14], where they use a second 'ScreenerNet' to assist a main network, integrating this second network in the loss function of the primary.

3.3 Results of Curriculum Learning

The implementation of Curriculum Learning has actually shown to have positive results on the training of a network. Bengio et al. [7] hypothesized that curriculum learning has an effect on the speed of convergence. In this case convergence is meant as the state of the neural network where the training has progressed enough to accurately respond to the training data provided with some defined margin of error. In certain cases, they discovered that easier sets of data could be more instructional than more difficult sets in earlier stages of the algorithms. They concluded that in their specific case they achieved better accuracy when working with curriculum learning, and finding a better local minima. Kerras et al. [13] uses a more implicit form of curriculum learning using a model approach, which progressively increases the amount of layers and in that way increases the networks capacity. Using the earlier resources to determine the structure of the samples before focusing it's resources for recognising details later. Hacohen et al. [5] evaluated multiple approaches, calculating the difference in confidence between a teacher-student network in a conventional way. They established that the teacher-student network had a faster training speed and allowed them to create new images of 'unprecedented quality'. They came to the conclusion that 'as long as the curriculum is positively correlated with the optimal utility, it can improve the learning'. Kim and Choi [14] take a different approach, using the student-teacher method to figure out the optimal weights for it's student network. It combines this with a self-paced strategy, using the loss of the student for optimal prediction of the weights of the network. Avramova [4] uses multiple variants of Self Paced Learning (SPL) and did find an increase in accuracy, because of the usage of a Curriculum Learning strategy.

Table 1. Results of Curriculum learning

Results of Curriculum Learning				
Paper	**Task**	**Method**	**Improvement**	**Dataset**
Bengio et al. [7]	Shape recognition	CL	*significantly better*	Geometric Shapes
Avramova [4]	Computer Vision	SPL/SPDL	~1%*	CIFAR-10
Kim et al.[14]	Computer Vision	Teacher-student	2%/ 0.15%	CIFAR-10/MNIST
Gong et al.[22]	Computer Vision	Teacher-student/CL (MMCL)	2-5%	Multiple (CIFAR100)
Jaing et al.[21]	Video Event Recognition	SPCL	*outperformance*	MED13/14Test
Tang et al.[23]	Computer Vision	SPDL	0.9% over ScSPM[26]	Caltech-101
Weinshall et al.[10]	Computer Vision	CL	~0.03%	CIFAR-100
Castells et al. [24]	Computer Vision	Dynamic-CL	0.45%	CIFAR-10/100
Qin et al.[25]	Computer Visions	Teacher-student (BLCL)	0.8-3.6%	CIFAR-10/100
Hacohen et al.[5]	Computer Vision	Teacher-student	0.5-1%	CIFAR-10/100
Li et al.[3]	Computer Vision	MICL	7.7%	VOC07

3.4 Age of acquisition

The Age of Acquisition is a theory and concept in language processing and acquisition, reflecting on the age and order of words learned in children and young adults ranging from the age of 1 to 25[27]. Words with a lower Age of Acquisition rating are usually shorter and more frequently appear during the neurological development of children. Brysbaert et al. [27] improved on the existing set of AoA ratings, by taking into account ambiguity of words. They updated the a database created in 1981 by validating AoA estimates and adding different sets of AoA research into the current set. This new set will be used as the ratings to base a curriculum on.

4 Methodology

This paper's contribution will focus on the effect of a curriculum, based on AoA, on a convolutional neural network. A combination of a pre-trained model (ResNet with ImageNet weights) together with the ILSVRC2017_CLS-LOC Imagenet dataset will be used to set a baseline and execute the experiment. To establish a difference in performance of the artificial intelligence, a baseline top-1 accuracy per class will be set. To study the effect of a curriculum in this particular case, the base of ResNet will be used with additional layers on top. These layers on top will consist of two dense (https://keras.io/api/layers/core_layers/dense/) layers, the first taking the output of the resnet model, using ReLu activation and outputting a (None,2048) shape. The second layer will function as the output classification layer and will therefore take the input of the last layer, and with an softmax activation output a (None, 1000) shape (The amount of classes in ImageNet). The code used for training can be found in *Appendix B*. The main focus of this paper lies on the effect of an Age of Acquisition based Curriculum, therefore, two primary experiments will be conducted, together forming a more clear image on how the Age of Acquisition for children could affect the training process of Neural networks.

4.1 Hypothesis

The main question to be answered in this section of the paper is:

Does a curriculum based on Age of Acquisition ratings positively influence generalisation and accuracy in convolutional neural networks?

This question will be answered using a supervised vanilla curriculum learning strategy, where the labels of difficulty are pre-defined by the researcher. I currently hypothesise that the main factors which have an effect on generalisation speed and accuracy do not necessarily coincide with elements that make a concept easier for us humans to understand. Meaning that I think a curriculum based on Age of Acquisition ratings, either low or high ratings, will not make a significant difference in generalisation or accuracy. To give an example to this train of thought; when Bengio et al[7] conducted the Geometric shapes experiment, they chose to implement more difficult images later on in the experiment (difficult meaning more complex, with more clutter and variety). This gave them significant results, whereas the classes to be picked by the Age of Acquisition rating do not necessarily conform to this order of complexity. To confirm this idea, this research will also feature an experiment which will feature two test with a curriculum based on accuracy from a baseline test. These test should provide more significant results if my hypothesis is correct, because these classes should contain images which should contain factors that have an greater effect on accuracy, and maybe generalisation.

4.2 ImageNet

The Dataset that is going to be used in this paper is the ImageNet Large Scale Visual Recognition Challenge database of 2017 [28], this is a dataset with the primary focus on being a benchmark in object category classification and detection. It contains 1000 classes of objects, with 1.3 million training images along with 100.000 test and 50.000 validation images. For the interest of this paper, only the object detection, not localisation, will be tested. This dataset was chosen for this paper as it has a great amount of classes, and variety between those classes. One of the earlier

datasets that was considered a possibility was COCo (https://cocodataset.org/home), but those 80 classes provided less freedom to accurately create an AoA-based curriculum. The ImageNet dataset provided ample freedom to map the AoA dataset onto the classes.

4.3 ResNet Baseline

The baseline will be created with a standard Resnet50 model, with weights trained on imagenet, and the dataset consisting of the validation set from the ILSVRC2017 dataset. Resnet50 is a variation of the ResNet model. It contains 48 convolutional layers together with 1 Max Pool and an Average Pool layer. The code for the validation of the created models is the same as the validation code in *Appendix E*, but with the model being:

```
model = tf.keras.applications.ResNet50(include_top=True, weights="imagenet",classes
    =1000, input_shape=None)
```

Each time a 'baseline' test or validation set is referenced, it will be a reference to this pre-trained resnet model and it's accuracy.

4.4 Age of Aquistition

The Age of Acquisition dataset that was chosen are the Test-based age-of-acquisition norms for 44 thousand English word meanings, created by Marc Brysbaert & Andrew Biemiller [27], based on a list from Dale and O'Rourke's Living Word Vocabulary. It contains 31.000 unique words and the corresponding AoA-Rating, for more information on this measure and information about how this rating came to be, I would recommend their paper. Of this set of 44 thousand words, 33.500 have an Age of Acquisition rating ranging from 1,6 (Mommy) to 25,0 (eisteddfod (being a musical contest)). Of these 33.500 words, 225 have a direct match with ImageNet Classes. For this research I decided against manually matching more of the ImageNet classes against the AoA set, the primary reason for this being the distinctness of some of the categories. To illustrate this; ImageNet has 5 classes which all feature a 'retriever' of some kind: *flat-coated retriever, curly-coated retriever, golden retriever, Labrador retriever, Chesapeake Bay retriever*. These could all be classified under 'retriever' with an AoA-rating of 8.7, but they could also be simplified down to 'dog' with an AoA-rating of 3.2, not even mentioning all the other types of dogs available. And the same is true for a multitude of animals. This pre-processing of the combination of the Age of Acquisition set and the ImageNet classes resulted in the total list which can be found in *Appendix B*, ranging from word like 'sock' (3.4) and 'cup' (3.4), to 'obelisk' (13.7) and 'bulbul' (17.2). From this table, the first and last 20 will be used as classes for training:

Table 2. Easiest AoA-rated ImageNet classes

ImageNet class	Class Name	AoA-rating
n04254777	sock	3,4
n02346627	cup	3,4
n02190166	fly	3,6
n07753592	banana	3,6
n02782093	balloon	3,7
n07747607	orange	3,8
n03938244	pillow	4
n02422106	bee	4
n07745940	strawberry	4,2
n03961711	plate	4,3
n02834397	bib	4,5
n09229709	bubble	4,5
n04371774	swing	4,5
n01608432	kite	4,6
n02799071	baseball	4,7
n03775071	mitten	4,7
n07697313	cheeseburger	4,8
n03814906	necklace	4,9
n01773549	barn	4,9
n01806143	peacock	4,9

Table 3. Hardest AoA-rated ImageNet classes

ImageNet class	Class Name	AoA-rating
n03680355	Loafer	11,9
n04147183	schooner	12,4
n02490219	marmoset	12,5
n01847000	drake	12,5
n04136333	sarong	12,9
n02091134	whippet	13,1
n03788195	mosque	13,2
n03297495	espresso	13,3
n04532670	viaduct	13,3
n03837869	obelisk	13,7
n04501370	turnstile	13,7
n02361337	marmot	13,9
n02389026	sorrel	14
n04141327	scabbard	14,1
n02011460	bittern	14,5
n02981792	catamaran	15,6
n02018795	bustard	15,8
n03146219	cuirass	15,9
n02006656	spoonbill	16
n01560419	bulbul	17,2

4.5 Approach

This research will feature 4 main tests. Two of them to visualise the difference in training between a curriculum of easy and hard classes based on the Age of Acquisition rating. The other two will function to get a closer look at the difference between classes with a higher and lower accuracy/-confidence (from the ResNet model). Next to these 4 training experiments, two 'validation' test will be conducted on randomly selected classes, these to verify the results from the earlier tests and provide a baseline for the difference in accuracy and loss during the training process. The chosen classes for high and low initial accuracy, and the randomly selected classes can be found in tables 4,5,6 and 7 respectively. All experiments will consist of training (*Appendix C*) with the specified classes, followed by using the generated model to determine the per class accuracy (*Appendix E*)

Table 4. Highest Accuracy classes (from ResNet)

ImageNet class	ImageNet Class Name	ResNet Accuracy
n02090622	borzoi	98
n02111129	Leonberg	98
n02342885	hamster	98
n01872401	echidna	98
n11939491	daisy	98
n12057211	yellow lady's slipper	98
n09288635	geyser	98
n01518878	ostrich	98
n01820546	lorikeet	98
n02006656	spoonbill	98
n02007558	flamingo	98
n06359193	web site	98
n13044778	earthstar	98
n02389026	sorrel	96
n02107683	Bernese mountain dog	96
n02489166	proboscis monkey	96
n02130308	cheetah	96
n03344393	fireboat	96
n12620546	hip	96
n11879895	rapeseed	96

Table 5. Lowest accuracy classes

ImageNet class	ImageNet Class Name	ResNet Accuracy
n03692522	loupe	26
n03476684	hair slide	26
n04286575	spotlight	26
n03045698	cloak	26
n01740131	night snake	24
n04008634	projectile	24
n04270147	spatula	24
n04264628	space bar	24
n03016953	chiffonier	20
n04356056	sunglasses	20
n03532672	hook	20
n04591157	Windsor tie	20
n03866082	overskirt	20
n03658185	letter opener	18
n02123159	tiger cat	16
n04152593	screen	16
n04355933	sunglass	16
n03710637	maillot	16
n04525038	velvet	14
n03832673	notebook	12

Table 6. Random ImageNet classes 1

ImageNet class	ImageNet Class Name	ResNet Accuracy
n01740131	night snake	24
n01742172	boa constrictor	76
n01744401	rock python	30
n07753592	banana	84
n04376876	syringe	56
n03761084	microwave	56
n03908618	pencil box	60
n02107908	Appenzeller	30
n02105855	Shetland sheepdog	60
n04033995	quilt	54
n07747607	orange	74
n04200800	shoe shop	68
n09288635	geyser	98
n04505470	typewriter keyboard	76
n02965783	car mirror	92
n09229709	bubble	78
n07831146	carbonara	74
n02102040	English springer	92
n02412080	ram	68
n04552348	warplane	76

Table 7. Random ImageNet classes 2

ImageNet class	ImageNet Class Name	ResNet Accuracy
n03838899	oboe	70
n04141076	sax	68
n03372029	flute	32
n11939491	daisy	98
n12057211	yellow lady's slipper	98
n09246464	cliff	62
n09468604	valley	82
n09193705	alp	50
n09472597	volcano	68
n09399592	promontory	44
n09421951	sandbar	58
n09256479	coral reef	82
n09332890	lakeside	54
n09428293	seashore	42
n09288635	geyser	98
n03498962	hatchet	50
n03393912	freight car	94
n03895866	passenger car	54
n02797295	barrow	74
n04204347	shopping cart	74

These six tables provide the basis for the experiments as the classes that will be used for training in each of the six examples. Each experiment will have two main outputs to be measured, the first is the training output, consisting of the training and validation loss, together with the training accuracy and validation accuracy for each epoch. The second output is the per class accuracy of the model created by each of the six training processes.

4.5.1 Limitations

Because of the size of the dataset (160Gb), it is not possible to upload the entirety or even a subset of it to a service like Google Collab (https://research.google.com/colaboratory/) which provides a lot more processing power for training these new layers. Therefore, the main limitation of this research was the specifications of the local machine on which the training was executed (Specs as found in DxDiag):

- **Motherboard:** MSI MPG X570 GAMING PLUS, AM4
- **Processor:** AMD Ryzen 7 3700X 8-Core (16 CPUs), 3.6GHz
- **Graphics Card:** NVIDIA GeForce RTX 2060
- **Memory:** 2x 8Gb - G.SKILL Trident Z RGB F4-3200C16D-16GTZR (16Gb - DDR4)
- **Operating System:** Windows 10 Pro 64-bit (10.0, Build 19042)
- **Storage:** Samsung SSD 970 EVO 1TB M.2 80mm

This system will run a WSL (Windows Subsystem for Linux) with Ubuntu 20.04 LTS from which python scripts will be ran.

These specifications, with the main constraint being the 16Gb of memory, means that there is a maximum to the amount of images that can be loaded and processed to be used for training. A heavy influence on this amount of images will be the batch size used in the training epochs. In general, smaller batch sizes could mean that the model can converge faster, but will train slower. With multiple experiments with the only variables being the total amount of images used for training and the batch size, a balance was struck on using 6000 images, with a batch size of 8. This however, bring with is another limitation. The ImageNet training dataset consists of 1.300 images per class. Using only 6000 images means that training on all classes with training weight assigned or oversampling would give results with an accuracy of less than 1.5% on all classes, with no significant difference in accuracy on oversampled or heavier weighed classes.

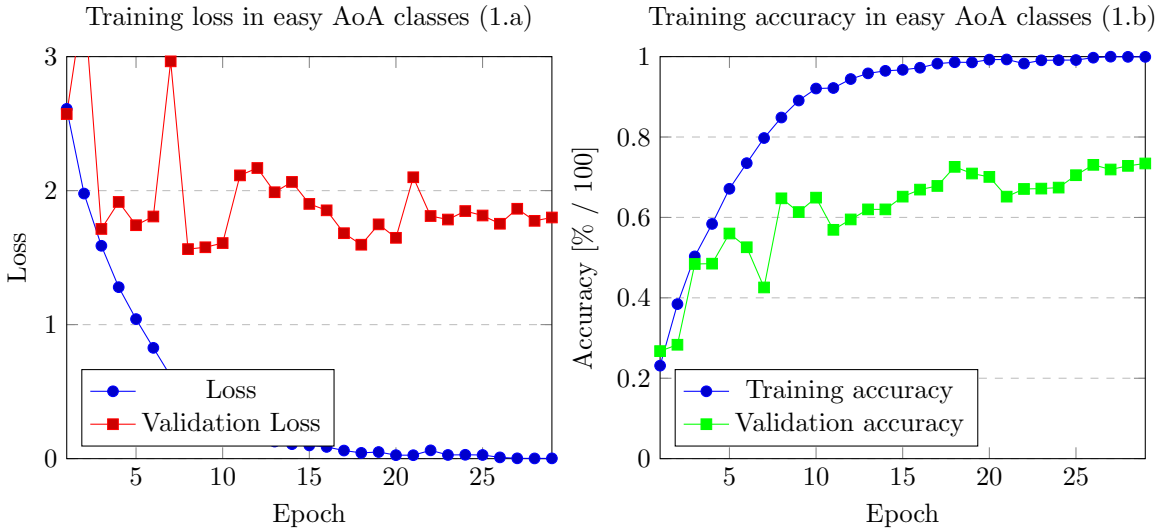
Hence the choice was made to solely train the top two added layers on the classes selected for that experiment. This will mean that this paper will not give the full answer to the question of how this version of curriculum learning affects the full training of a neural networks, but it will give more insight into the effect of training on different classes and their outcomes.

5 Results

5.1 Training process

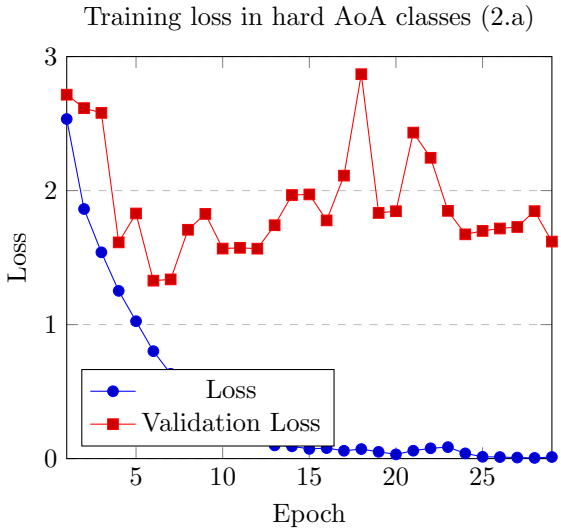
The training process consisted of using the code in *Appendix D* set to 29 epochs, which was chosen after multiple test showed no more than 2% increase on top of the values set in epoch 29 . Each epoch on average finished after 737 seconds (12m17s). Each experiment will feature two tables of output (*Appendix E*) split into three graphs. The first two graphs (a & b) will be featured in this subsection, and will feature the loss (a) and accuracy (b) during the training process. The last graph (c) will feature the validation accuracy on each individual class that was trained on, in comparison to the baseline ResNet model.

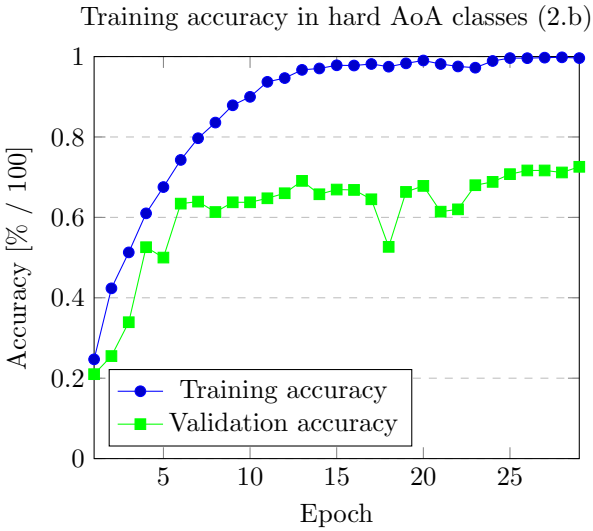
5.1.1 Easy AoA



The model based on the easiest classes in the Age of Acquisition set achieved the lowest training loss on epoch 28 with 0.0017. It's accuracy was the highest on epoch 27 with 0.9998. It's validation loss was the lowest on epoch 8 with 1.5636, and it achieved a maximum validation accuracy of 0.7342 on epoch 29.

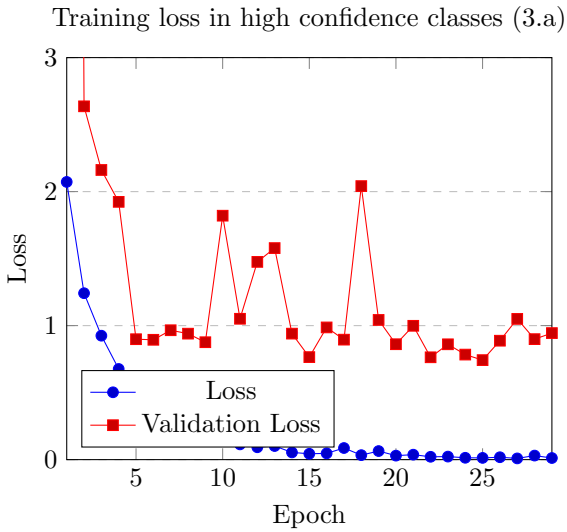
5.1.2 Hard AoA

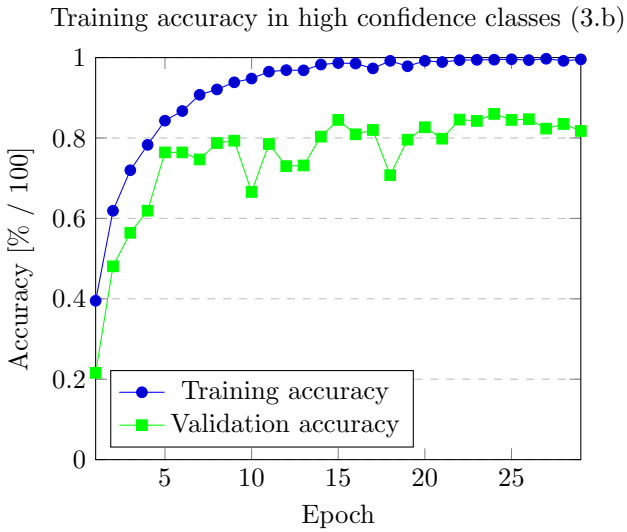




The model based on the hardest classes in the Age of Acquisition set achieved the lowest training loss on epoch 27 with 0.0077. It's accuracy was the highest on epoch 28 with 0.9981. It's validation loss was the lowest on epoch 7 with 1.3373, and it achieved a maximum validation accuracy of 0.7258 on epoch 29, which is 0.0084 lower than the Easy AoA model.

5.1.3 High confidence (High CI)

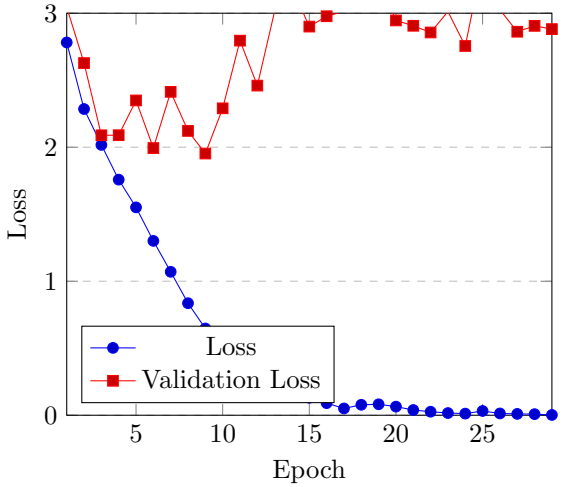




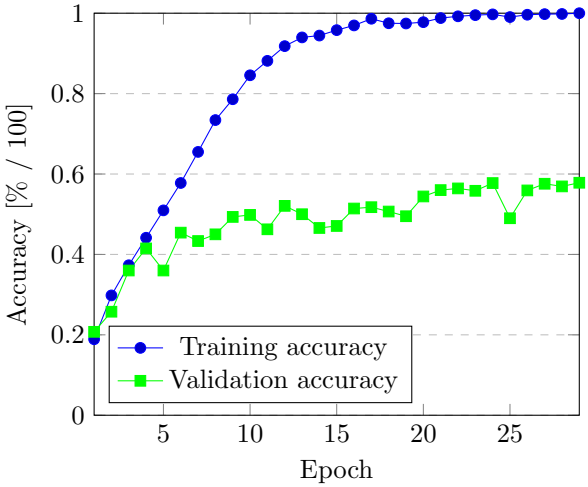
The model based on the highest confidence classes from the baseline achieved the lowest training loss on epoch 29 with 0.0012. It's accuracy was the highest on epoch 27 with 0.9971. It's validation loss was the lowest of all experiments on epoch 25 with 0.7425, and it achieved a maximum validation accuracy of all experiments of 0.86 on epoch 24.

5.1.4 Low confidence (Low CI)

Training loss in low confidence classes (4.a)



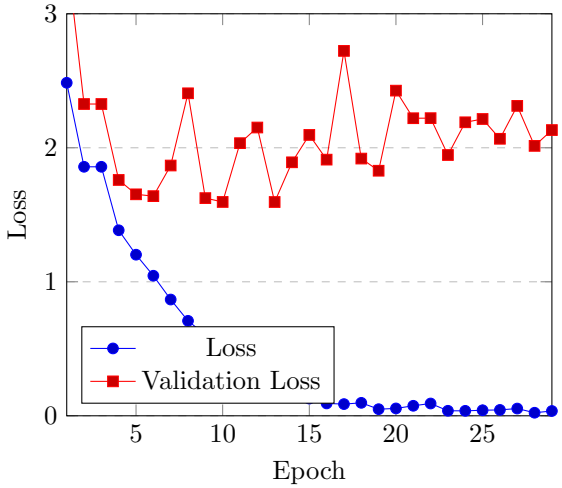
Training accuracy in low confidence classes (4.b)



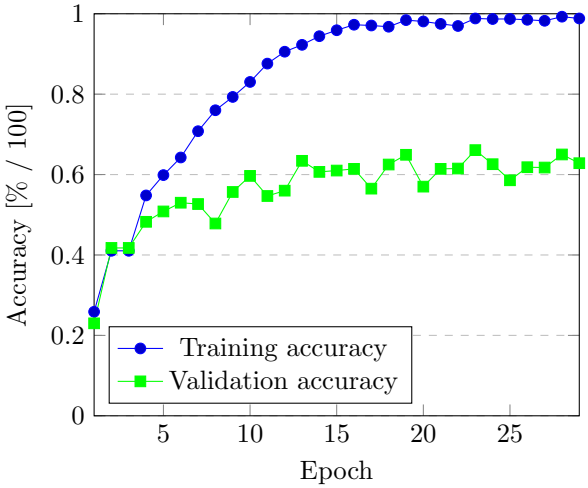
The model based on the lowest confidence classes from the baseline achieved the lowest training loss of all experiments on epoch 27 with 0.001. It's accuracy was the highest off all experiments on epoch 29 with 1. It's validation loss was the lowest on epoch 9 with 1.9529, and it achieved a maximum validation accuracy of 0.5783 on epoch 29.

5.1.5 Random Categories 1

Training loss in random classes (5.a)



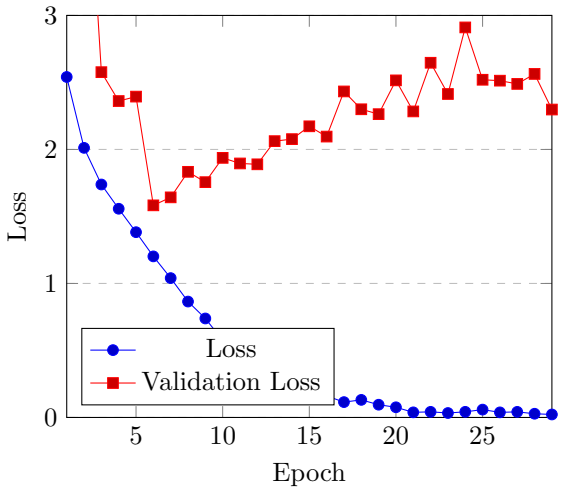
Training accuracy in random classes (5.b)



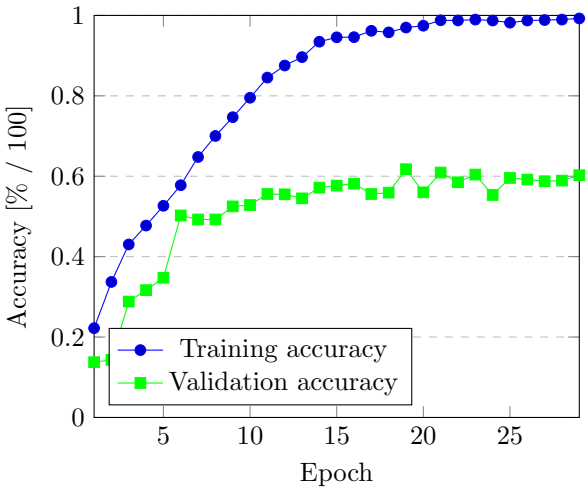
The first model trained on randomly selected classes achieved the lowest training loss on epoch 28 with 0.0228. It's accuracy was the highest on epoch 28 with 0.9927. It's validation loss was the lowest on epoch 13 with 1.5953, and it achieved a maximum validation accuracy of 0.6608 on epoch 23.

5.1.6 Random Categories 2

Training loss in random classes 2 (6.a)



Training accuracy in random classes 2 (6.b)



The second model trained on randomly selected classes achieved the lowest training loss on epoch 29 with 0.0223. It's accuracy was the highest on epoch 29 with 0.9925. It's validation loss was the lowest on epoch 6 with 1.5829, and it achieved a maximum validation accuracy of 0.6175 on epoch 19. Below are all the maximum scores the models achieved during training:

Table 8. Maximum scores from the training process of all models

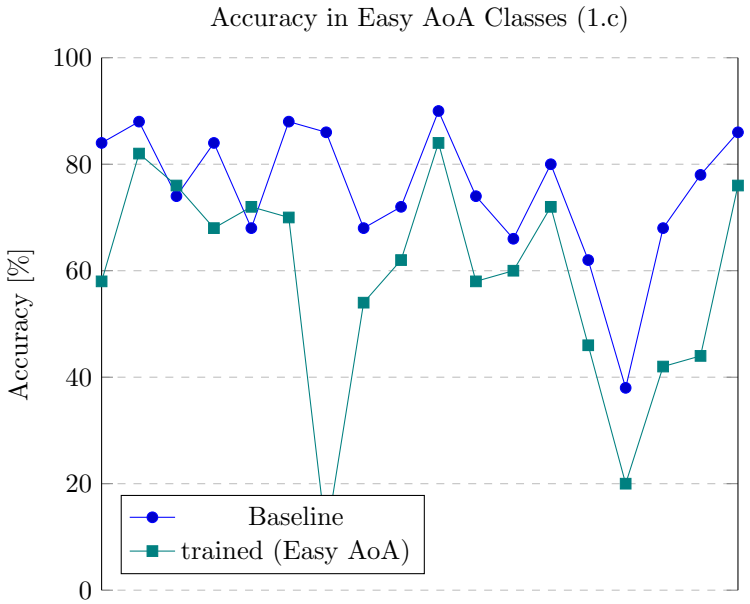
Max Scores	Loss		Accuracy		Val_loss		Val_accuracy	
	Epoch	Loss	Epoch	Accuracy	Epoch	Val_loss	Epoch	Val_accuracy
Easy AoA	28	0.0017	27	0.9998	8	1.5636	29	0.7342
Hard AoA	27	0.0077	28	0.9981	7	1.3373	29	0.7258
High CI	29	0.0012	27	0.9971	25	0.7425	24	0.86
Low CI	27	0.001	29	1	9	1.9529	29	0.5783
Random	28	0.0228	28	0.9927	13	1.5953	23	0.6608
Random2	29	0.0223	29	0.9925	6	1.5829	19	0.6175

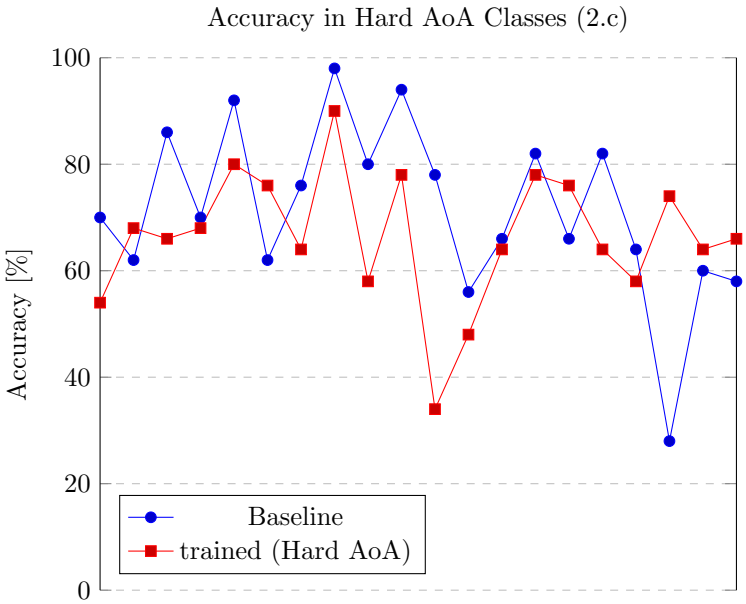
The most interesting/remarkable item in this table is the High CI Validation Loss, both because it is high in Epoch, but also because it is by far the lowest validation loss achieved by any model. Furthermore, the loss of both the Random trained models, as they are far higher than the curriculum trained models and almost equal. Next to that, it is noteworthy that 3/4 curriculum trained models outperform the random trained models, the only model not to achieve a higher validation accuracy is the model trained only on the 'hardest' classes in ImageNet (classes with the lowest

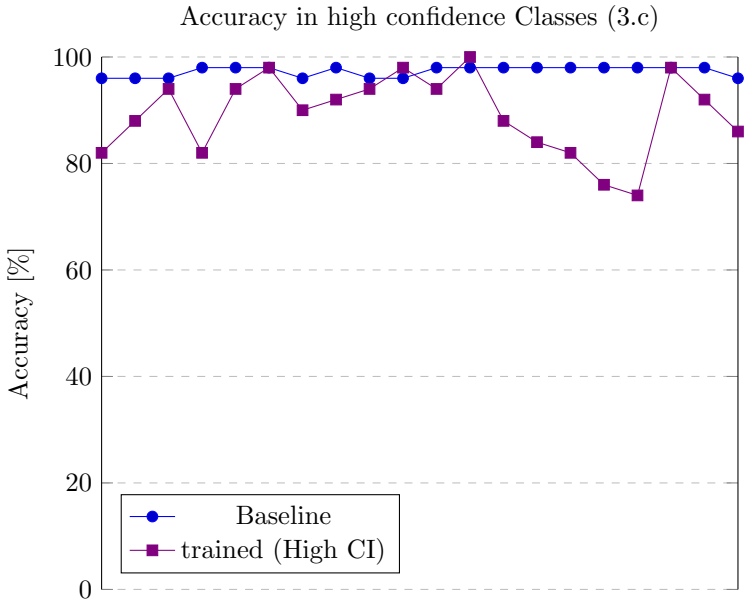
baseline accuracy) but which on the other hand showed the best increase in accuracy from the baseline.

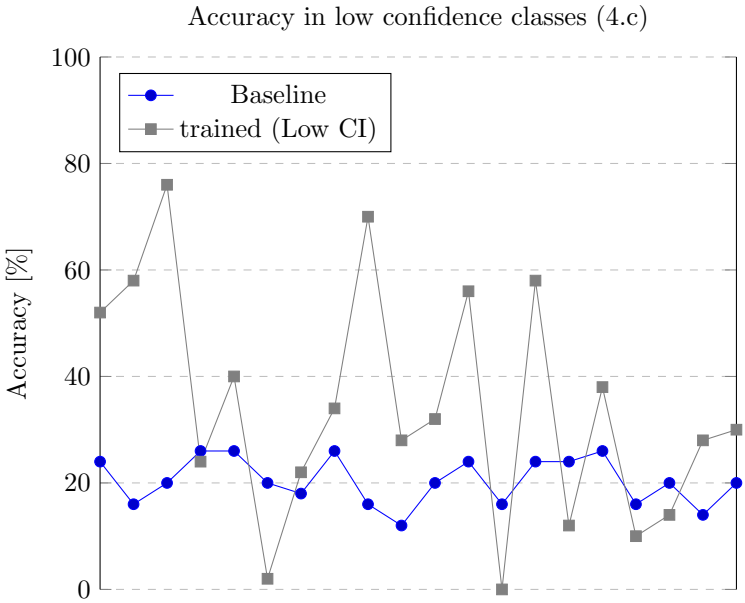
5.2 Accuracy

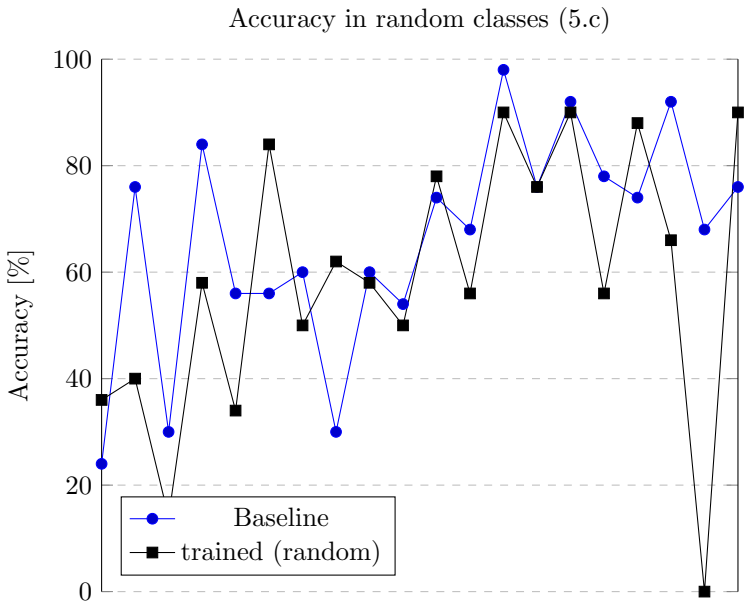
This section covers the accuracy of the baseline model in comparison to the trained models. The first graphs will provide an overview of the accuracy per class of each of the 20 classes used for training (1.c,2.c,3.c,4.c) for each curriculum, together with the accuracy of the random classes (5.c,6.c).











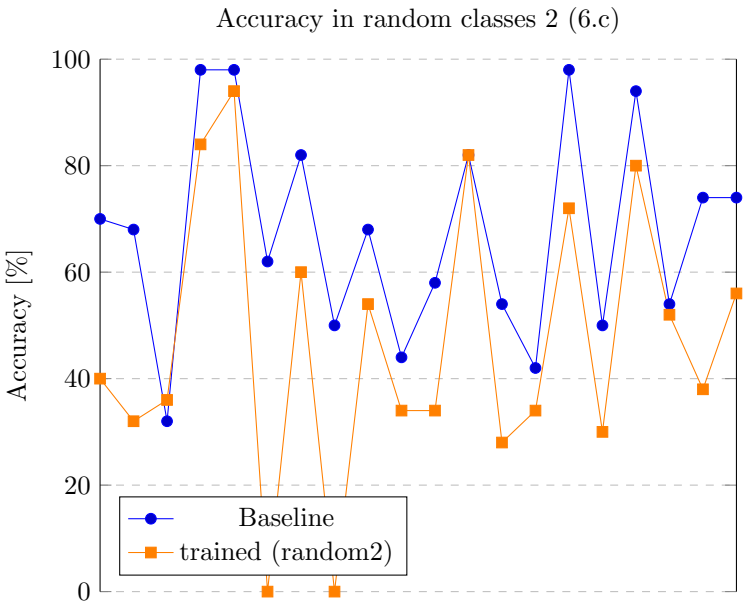


Table 9. Average accuracy of curriculum baseline and trained models

Curriculum	Baseline average accuracy	Trained average accuracy	Difference in accuracy
Easy AoA	75.22	58.55	-16.67
hard AoA	69.8	66.4	-3.4
high CI	97.3	85	-12.3
low CI	20.4	34.2	13.8
Random	66.3	58.8	-7.5
Random2	67.6	47	-20.6

The biggest outlier in this section of the experiment is the low confidence classes, the trained average is still much lower than those of the randomly chosen classes or even the AoA classes for that matter, but it is also the only curriculum where the average accuracy increased from the baseline to the trained model. Also notable is the big difference in trained average accuracy in the two random classes with repesct to the baseline average. While the baselines only differ 1.3%, the trained models have a difference of 11.8%. The difference between the baseline averages of the AoA classes comes down to (75.22 - 69.8 =) 5.42%, and the trained averages differ (58.55 - 66.4 =) -7.85%.

6 Discussion

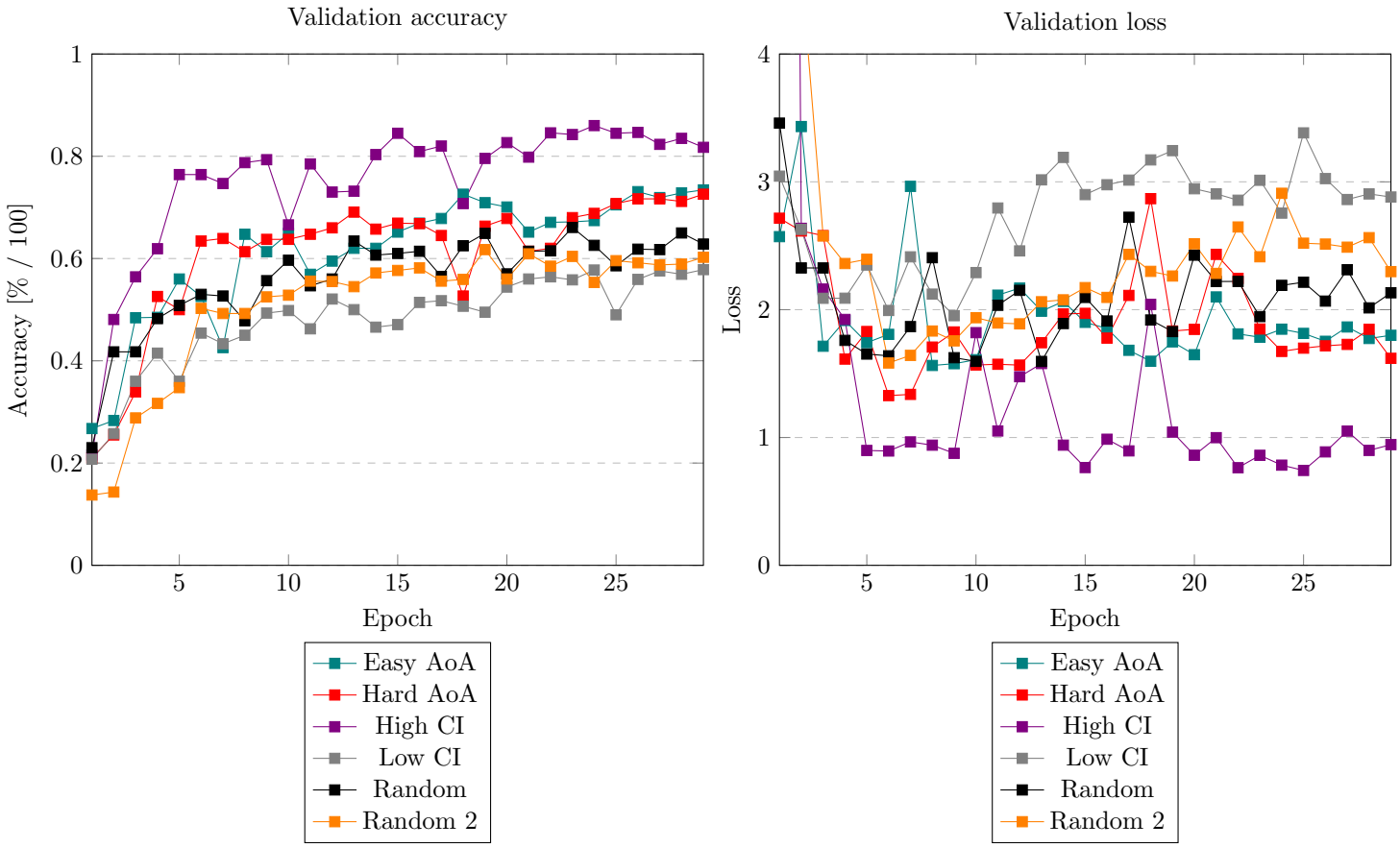
6.1 Results

The training process results as described in table 7 show the most promising results, in this data, there is a distinct difference in the validation accuracy of both the AoA-based curriculum models (0.73422 & 0.7258) and the randomly chosen models (0.6608 & 0.6175). However this difference does not seem to be reflected in the average accuracy of the trained models (58.55 & 66.4 for the AoA models and 58.8 & 47 for the random models. On the other hand, this could be explained by the low amount of images used in training; Because this quantity was limited to 6000, only 23.07% of the 26.000 available images (20 * 1.300), meaning only about 240 (1.300*23%*0.8) images per class were used for training (20 per cent of images were used as the validation split (60)). This could mean that some features found in the 1000 other not-used images would not be analysed during training meaning the validation accuracy of the 50 val images could differ drastically. Because of this limitation, I am hesitant to draw conclusions from the accuracy of the trained models, as multiple factors, not just the Age of Acquisition could have had an effect on the training process, especially because the differences in accuracy between the random classes and the AoA classes are also much higher than expected and seen in the validation accuracy during training. The validation accuracy during training however do show a significant difference, 0.7342 and 0.7258 for the AoA trained models, while the random trained models sit at 0.6608 and 0.6175. This disparity could mean that the AoA-classes do contain qualities which improve training validation accuracy. The training accuracy of these models also differ, with 0.9998 and 0.9981 to the randomly trained 0.9927 and 0.9925, but only with an average of about 0.64% (against the validation accuracy's 9.08%) The curriculum's based on the initial confidence of the ResNet model show that factors other than the Age of Acquisition seem to have a bigger effect on the training process. The high CI curriculum has by far the lowest validation loss and highest validation accuracy of all of the models, whereas the low CI curriculum had the highest validation loss and the lowest validation accuracy. This in my opinion could be the result of one of two factors.

1. The training images for the high confidence classes have a higher similarity to their validation counterparts than normal or randomly selected classes, and the low confidence's validation images are the most different from their training versions. Meaning the training process is more efficient for these classes.

2. The high confidence images are easier for the network to recognise features in, e.g. the clutter in these images is lower, or the shape distinctiveness is more prevalent, etc. And the opposite would be true for the lower confidence classes.

The high confidence training validation accuracy is the first to reach a value within 5% of it's final score (0.776625) on epoch 9, and the low confidence training is the last to achieve a value within 5% (Easy AoA: 18, Hard AoA: 13, High CI: 9, Low CI: 21, Random: 13, Random2: 15). This shows that the high confidence model seems to converge faster than any other model, whereas the low confidence model is the slowest to converge, and would therefore need the most amount of training time to get to a model which could generalise.



In both validation metrics, the High CI stands out as the main outlier, almost fully outperforming all other models on both cases. Both AoA cases also almost seem to conform to each other, both outperforming the randomly chosen categories.

6.2 Hypothesis

While the limitations of this research could influence the results, both AoA curriculum models do exceed the performance of the models without a curriculum, in validation accuracy during training and in validaiton loss during training. However, apart from the difference in validation accuracy of the trained model on the validation set, both AoA-based curriculum models perform almost equal. While I am hesitant to attribute the success of these AoA-based curriculum models to the selection of classes based on Age of Acquisition, the models in this particular scenario do perform better. However, the biggest point of uncertainty is that, as a result of the limitations, only the performance on the selected classes was measured. This raises the question whether the improvements found in the AoA-based curriculum models would also apply to an over-sampled or weighed training setup. Which is one of the main points of hesitation with these results in comparison to the hypothesis.

6.3 Further Work

To fortify these results, this experiment absolutely needs to be repeated on more datasets, and on a larger scale. The limitation of the researcher's hardware could have hurt this experiment more than can be researched at the moment. Repeating this experiment on different datasets could sketch a better understanding on the variation in classes which are both low in AoA-rating, but also perform better. To validate if these results also hold true when only a slight bias or oversampling is given to the chosen classes, a larger experiment is needed.

References

[1] Jiang, Yang and Bosch, Nigel and Baker, Ryan and Paquette, Luc and Ocumpaugh, Jaclyn and Andres, Alexandra and Moore, Allison and Biswas, Gautam.: *Expert Feature-Engineering vs. Deep Neural Networks: Which Is Better for Sensor-Free Affect Detection?* (Nov 2018)

[2] LeCun, Y., Bengio, Y., Hinton, G.: *Deep Learning. Nature.* 521, 436–444 (2015)

[3] Siyang Li, Xiangxin Zhu, Qin Huang, Hao Xu and C.-C. Jay Kuo.: *Multiple Instance Curriculum Learning for Weakly Supervised Object Detection* (Nov 2017)

[4] Vanya Avramova.: *Curriculum Learning with Deep Convolutional Neural Networks* (2015)

[5] Guy Hacohen & Daphna Weinshall, "*On the power of curriculum learning in training deep networks*," in Proceedings of ICML, vol. 97, 2019

[6] Xuan Zhang , Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat.: *An Empirical Exploration of Curriculum Learning for Neural Machine Translation* (Nov 2018)

[7] Yoshua Bengio, Jérôme Louradour, Ronan Collobert and Jason Weston.: *Curriculum learning.* In: ICML (2009)

[8] Petru Soviany, Radu Tudor Ionescu, Paolo Rota and Nicu Sebe.: *Curriculum Learning: A Survey* (Jan 2021)

[9] Takayoshi Yamashita and Taro Watasue.: *Hand posture recognition based on bottom-up structured deep convolutional neural network with curriculum learning* (Jan 2015)

[10] Daphna Weinshall, Gad Cohen and Dan Amir.: *Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks* In: ICML (June 2018)

[11] Gustavo Penha and Claudia Hauff.: *Curriculum Learning Strategies for IR* In: Jose J. et al. (eds) Advances in Information Retrieval. ECIR. Lecture Notes in Computer Science, vol 12035. Springer, Cham. (April 2020)

[12] Yiru Wang, Weihao Gan, Jie Yang, Wei Wu, Junjie Yan; *Proceedings of the IEEE/CVF International Conference on Computer Vision* (ICCV), (2019)

[13] T. Karras, T. Aila, S. Laine, and J. Lehtinen, *"Progressive Growing of GANs for Improved Quality, Stability, and Variation,"* in Proceedings of ICLR, (2018)

[14] T.-H. Kim and J. Choi, *"Screenernet: Learning self-paced curriculum for deep neural networks,"* arXiv preprint arXiv:1801.00904, (2018)

[15] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann, *"Selfpaced curriculum learning."* in Proceedings of AAAI, vol. 2, (2015)

[16] Avrahami, J., Kareev, Y., Bogot, Y., Caspi, R., Dunaevsky, S., and Lerner, S. *Teaching by examples: Implications for the process of category acquisition.* The Quarterly Journal of Experimental Psychology: Section A, 50(3): 586–606, 1997.

[17] Elman, J. L. (1993). *Learning and development in neural networks: The importance of starting small*. Cognition, 48, 781–799.

[18] R. T. Ionescu, B. Alexe, M. Leordeanu, M. Popescu, D. P. Papadopoulos, and V. Ferrari, *"How hard can it be? estimating the difficulty of visual search in an image,"* in Proceedings of CVPR, 2016, pp. 2157– 2166

[19] X. Zhang, G. Kumar, H. Khayrallah, K. Murray, J. Gwinnup, M. J. Martindale, P. McNamee, K. Duh, and M. Carpuat, *"An empirical exploration of curriculum learning for neural machine translation,"*, (2018)

[20] A. Kamilaris, X. Francesc, Prenafeta-Boldú, "*Deep Learning in agriculture: A survey*", Computers and Electronics in Agriculture, Volume 147, (2018)

[21] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann, "*Selfpaced curriculum learning.*" in Proceedings of AAAI, vol. 2, 2015, p. 6.

[22] C. Gong, D. Tao, S. J. Maybank, W. Liu, G. Kang, and J. Yang, "*Multimodal curriculum learning for semi-supervised image classification*," IEEE Transactions on Image Processing, vol. 25, no. 7, pp. 3249– 3260, 2016

[23] Y. Tang, Y.-B. Yang, and Y. Gao, "*Self-paced dictionary learning for image classification*," in Proceedings of ACMMM, 2012, pp. 833–836.

[24] T. Castells, P. Weinzaepfel, and J. Revaud, "*Superloss: A generic loss for robust curriculum learning*," Proceedings of NIPS, vol. 33, 2020

[25] W. Qin, Z. Hu, X. Liu, W. Fu, J. He, and R. Hong, "*The balanced loss curriculum learning*," IEEE Access, vol. 8, pp. 25 990–26 001, 2020.

[26] J. Yang, K. Yu, Y. Gong, and T. Huang. "*Linear spatial pyramid matching using sparse coding for image*", classification. In CVPR, pages 1794 – 1801, 2009.

[27] Brysbaert, M., Biemiller, A. Test-based age-of-acquisition norms for 44 thousand English word meanings. Behav Res 49, 1520–1523 (2017). https://doi.org/10.3758/s13428-016-0811-4

[28] Olga Russakovsky and Jia Deng and Hao Su and Jonathan Krause and Sanjeev Satheesh and Sean Ma and Zhiheng Huang and Andrej Karpathy and Aditya Khosla and Michael Bernstein and Alexander C. Berg and Li Fei-Fei. *"ImageNet Large Scale Visual Recognition Challenge"*. International Journal of Computer Vision (IJCV). 2015. vol. 115. pages 211-252.

A Full data

A.1 Easy AoA

Table 10.a Easy AoA training

Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1	2.61	0.2313	2.5714	0.2675
2	1.9781	0.3844	3.4327	0.2833
3	1.5884	0.5031	1.7141	0.4842
4	1.2796	0.5838	1.9155	0.485
5	1.0413	0.6712	1.7419	0.56
6	0.8267	0.735	1.8069	0.5258
7	0.6151	0.7975	2.9648	0.4258
8	0.4651	0.8485	1.5636	0.6475
9	0.331	0.8908	1.5773	0.6133
10	0.2415	0.9206	1.6087	0.6492
11	0.2449	0.9219	2.1137	0.5692
12	0.1742	0.9442	2.1688	0.595
13	0.1274	0.9583	1.988	0.62
14	0.108	0.9646	2.0649	0.62
15	0.0988	0.9671	1.9009	0.6517
16	0.0875	0.9723	1.853	0.6692
17	0.0604	0.9825	1.6823	0.6783
18	0.0431	0.986	1.5964	0.7258
19	0.0494	0.9856	1.7483	0.7092
20	0.0261	0.9925	1.6478	0.7008
21	0.0257	0.9929	2.0997	0.6517
22	0.062	0.9827	1.8105	0.6708
23	0.0275	0.9908	1.7843	0.6717
24	0.0288	0.9912	1.8475	0.6742
25	0.0273	0.9915	1.8144	0.705
26	0.0094	0.9973	1.7528	0.7308
27	0.0024	0.9998	1.8647	0.7192
28	0.0017	0.9994	1.7746	0.7283
29	0.0021	0.9994	1.7995	0.7342

Table 10.b Easy AoA accuracy

Class	Baseline	Trained
n07745940	84	58
n02782093	88	82
n07747607	74	76
n07753592	84	68
n01608432	68	72
n01806143	88	70
n01828970	86	10
n04371774	68	54
n02190166	72	62
n03814906	90	84
n02799071	74	58
n03775071	66	60
n03938244	80	72
n02834397	62	46
n02895154	38	20
n04254777	68	42
n09229709	78	44
n07697313	86	76

A.2 Hard AoA

Table 11.a Hard AoA training

Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1	2.5339	0.2469	2.7155	0.21
2	1.8625	0.4235	2.6153	0.255
3	1.5393	0.5129	2.5798	0.3392
4	1.2516	0.61	1.6135	0.5258
5	1.0252	0.6754	1.8292	0.5
6	0.8016	0.7429	1.3278	0.6342
7	0.6325	0.7969	1.3373	0.6392
8	0.4881	0.8358	1.7075	0.6133
9	0.3694	0.8788	1.8259	0.6375
10	0.2991	0.8998	1.5673	0.6375
11	0.2059	0.9371	1.5729	0.6475
12	0.164	0.9465	1.5664	0.66
13	0.0986	0.9669	1.7416	0.6908
14	0.0918	0.9704	1.9666	0.6575
15	0.0724	0.9783	1.9714	0.6692
16	0.0784	0.9777	1.7775	0.6683
17	0.0584	0.9815	2.1115	0.645
18	0.0711	0.9748	2.8685	0.5267
19	0.0505	0.9831	1.8333	0.6633
20	0.0315	0.9902	1.846	0.6783
21	0.0588	0.9815	2.4327	0.6142
22	0.0769	0.9754	2.2447	0.62
23	0.0858	0.9723	1.8488	0.68
24	0.039	0.989	1.6736	0.6883
25	0.0136	0.9962	1.699	0.7075
26	0.0115	0.996	1.7171	0.7167
27	0.0077	0.9973	1.7284	0.7167
28	0.0051	0.9981	1.8465	0.7117
29	0.0119	0.9962	1.6195	0.7258
30	0.0229	0.9929	1.9447	0.6983

Table 11.b Hard AoA accuracy

Class	Baseline	Trained
n02442845	70	54
n02113186	62	68
n02454379	86	66
n01883070	70	68
n03447447	92	80
n03868242	62	76
n02804610	76	64
n09288635	98	90
n01582220	80	58
n01592084	94	78
n01797886	78	34
n04376876	56	48
n01943899	66	64
n03781244	48	78
n02859443	66	76
n04277352	82	64
n03527444	64	58
n03782006	28	74
n02790996	60	64
n03443371	58	66

A.3 High CI

Table 12.a High CI training

Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1	2.0723	0.3952	26.1011	0.2158
2	1.2419	0.619	2.6365	0.4808
3	0.9248	0.7198	2.1614	0.5642
4	0.6765	0.7829	1.9236	0.6192
5	0.502	0.8431	0.8986	0.7642
6	0.4129	0.8671	0.8943	0.7642
7	0.2933	0.9075	0.9655	0.7467
8	0.249	0.9208	0.94	0.7875
9	0.1896	0.9385	0.8761	0.7933
10	0.1576	0.9479	1.8203	0.6658
11	0.1137	0.965	1.0511	0.785
12	0.0922	0.9688	1.4756	0.73
13	0.1014	0.9683	1.5785	0.7317
14	0.0538	0.9827	0.9402	0.8033
15	0.0437	0.9865	0.765	0.845
16	0.0467	0.9852	0.9866	0.8092
17	0.0866	0.9731	0.8953	0.82
18	0.0335	0.9919	2.0411	0.7075
19	0.0639	0.9785	1.0422	0.7958
20	0.0298	0.9919	0.8611	0.8267
21	0.0369	0.989	0.9987	0.7983
22	0.0212	0.9937	0.7637	0.8458
23	0.0226	0.9944	0.8607	0.8425
24	0.0138	0.9948	0.7838	0.86
25	0.0129	0.9958	0.7425	0.845
26	0.0179	0.9937	0.8874	0.8467
27	0.0091	0.9971	1.0501	0.8233
28	0.0302	0.9919	0.8991	0.835
29	0.012	0.9956	0.9443	0.8175

Table 12.b High CI accuracy

Class	Baseline	Trained
n02389026	96	82
n02107683	96	88
n02489166	96	94
n02090622	98	82
n02111129	98	94
n02342885	98	98
n02130308	96	90
n01872401	98	92
n03344393	96	94
n11879895	96	98
n11939491	98	94
n12057211	98	100
n09288635	98	88
n01518878	98	84
n01820546	98	82
n02006656	98	76
n02007558	98	74
n06359193	98	98
n13044778	98	92
n12620546	96	0

A.4 Low CI

Table 13.a Low CI training

Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1	2.7819	0.189	3.0447	0.2075
2	2.2851	0.2981	2.628	0.2575
3	2.0173	0.3731	2.0891	0.36
4	1.7581	0.4415	2.0898	0.415
5	1.5512	0.5096	2.3492	0.36
6	1.301	0.5777	1.9938	0.4542
7	1.0703	0.655	2.4133	0.4333
8	0.8361	0.7342	2.1217	0.45
9	0.6474	0.786	1.9529	0.4933
10	0.4507	0.8456	2.2903	0.4983
11	0.3486	0.8815	2.7954	0.4625
12	0.2343	0.9181	2.4598	0.5208
13	0.1856	0.9398	3.0156	0.5
14	0.1673	0.9444	3.1913	0.4658
15	0.1302	0.9579	2.9002	0.4708
16	0.0903	0.9696	2.9774	0.5142
17	0.0507	0.9862	3.0137	0.5175
18	0.078	0.975	3.1717	0.5067
19	0.0818	0.9744	3.2444	0.495
20	0.0643	0.9777	2.9461	0.5442
21	0.039	0.9879	2.9063	0.56
22	0.0263	0.9923	2.8563	0.5642
23	0.0164	0.9952	3.0124	0.5583
24	0.0121	0.9971	2.7555	0.5775
25	0.0318	0.9904	3.3832	0.49
26	0.0134	0.9962	3.0255	0.5592
27	0.01	0.9977	2.8618	0.5758
28	0.0081	0.9981	2.9056	0.5692
29	0.0022	1.0000	2.8814	0.5783

Table 13.b Low CI accuracy

Class	Baseline	Trained
n03692522	26	34
n03476684	26	40
n04286575	26	38
n03045698	26	24
n01740131	24	52
n04008634	24	56
n04270147	24	12
n04264628	24	58
n03016953	20	76
n04356056	20	14
n03532672	20	2
n04591157	20	30
n03866082	20	32
n03658185	18	22
n02123159	16	58
n04152593	16	0
n04355933	16	10
n03710637	16	70
n04525038	14	28
n03832673	12	28

A.5 Random

Table 14.a Random classes training

Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1	2.485	0.2587	3.4595	0.23
2	1.8581	0.4104	2.3265	0.4175
3	1.8581	0.4104	2.3265	0.4175
4	1.3845	0.5481	1.7598	0.4825
5	1.2021	0.5987	1.6527	0.5083
6	1.0452	0.6425	1.6391	0.53
7	0.8675	0.7079	1.8676	0.5267
8	0.7085	0.76	2.4066	0.4783
9	0.6043	0.7931	1.6244	0.5567
10	0.4878	0.8304	1.5957	0.5967
11	0.367	0.876	2.0344	0.5467
12	0.2761	0.9056	2.1512	0.56
13	0.2172	0.9227	1.5953	0.6342
14	0.1682	0.9442	1.8916	0.6067
15	0.1284	0.959	2.0958	0.61
16	0.0923	0.9727	1.9115	0.6142
17	0.0873	0.9708	2.7229	0.565
18	0.0967	0.9677	1.9194	0.625
19	0.0494	0.9842	1.8287	0.6492
20	0.0549	0.9808	2.4264	0.57
21	0.0747	0.9748	2.2205	0.6142
22	0.0917	0.9696	2.221	0.615
23	0.0374	0.9883	1.946	0.6608
24	0.037	0.9869	2.1902	0.6258
25	0.0414	0.9871	2.2147	0.5858
26	0.0439	0.985	2.0671	0.6183
27	0.054	0.9825	2.3125	0.6175
28	0.0228	0.9927	2.0141	0.65
29	0.0358	0.9883	2.1322	0.6283

Table 14.b Random Classes accuracy

Class	Baseline	Trained
n01740131	24	36
n01742172	76	40
n01744401	30	14
n07753592	84	58
n04376876	56	34
n03761084	56	84
n03908618	60	50
n02107908	30	62
n02105855	60	58
n04033995	54	50
n07747607	74	78
n04200800	68	56
n09288635	98	90
n04505470	76	76
n02965783	92	90
n09229709	78	56
n07831146	74	88
n02102040	92	66
n02412080	68	0
n04552348	76	90

A.6 Random 2

Table 15.a Random classes training 2

Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1	2.5408	0.2221	7.5304	0.1375
2	2.0118	0.3371	4.6658	0.1433
3	1.7384	0.4304	2.5767	0.2883
4	1.5574	0.4773	2.3611	0.3167
5	1.3826	0.5265	2.3943	0.3475
6	1.202	0.5777	1.5829	0.5025
7	1.0399	0.6479	1.6425	0.4925
8	0.8649	0.7002	1.833	0.4925
9	0.7387	0.7469	1.7559	0.525
10	0.5755	0.7952	1.9372	0.5283
11	0.4519	0.8454	1.8955	0.5558
12	0.3621	0.8754	1.8891	0.555
13	0.2934	0.8963	2.062	0.545
14	0.1942	0.9346	2.077	0.5717
15	0.1686	0.9456	2.1727	0.5767
16	0.1607	0.946	2.096	0.5817
17	0.1146	0.9617	2.4329	0.5558
18	0.1312	0.9579	2.2998	0.5592
19	0.0951	0.9698	2.2633	0.6175
20	0.076	0.9744	2.5155	0.56
21	0.0382	0.9879	2.2838	0.6092
22	0.042	0.9875	2.6469	0.585
23	0.0332	0.9896	2.4138	0.6042
24	0.0421	0.9873	2.9106	0.5533
25	0.0586	0.9819	2.52	0.5958
26	0.038	0.9873	2.5132	0.5917
27	0.0418	0.9885	2.489	0.5875
28	0.0278	0.9896	2.5635	0.5892
29	0.0223	0.9925	2.2978	0.6025

Table 15.b Random classes 2 accuracy

Class	Baseline	Trained
n03838899	70	40
n04141076	68	32
n03372029	32	36
n11939491	98	84
n12057211	98	94
n09246464	62	0
n09468604	82	60
n09193705	50	0
n09472597	68	54
n09399592	44	34
n09421951	58	34
n09256479	82	82
n09332890	54	28
n09428293	42	34
n09288635	98	72
n03498962	50	30
n03393912	94	80
n03895866	54	52
n02797295	74	38
n04204347	74	56

B AoA mapped to ImageNet with AoA-rating

ImageNet class	Class Name	AoA-rating
n04254777	sock	3,4
n02346627	cup	3,4
n02190166	fly	3,6
n07753592	banana	3,6
n02782093	balloon	3,7
n07747607	orange	3,8
n03938244	pillow	4
n02422106	bee	4
n07745940	strawberry	4,2
n03961711	plate	4,3
n02834397	bib	4,5
n09229709	bubble	4,5
n04371774	swing	4,5
n01608432	kite	4,6
n02799071	baseball	4,7
n03775071	mitten	4,7
n07697313	cheeseburger	4,8
n03814906	necklace	4,9
n01773549	barn	4,9
n01806143	peacock	4,9
n02137549	goose	4,9
n03661043	library	5
n12267677	corn	5
n04592741	wing	5
n04127249	safe	5,1
n07749582	lemon	5,1
n02980441	castle	5,1
n07714990	broccoli	5,2
n01514859	hen	5,2
n04133789	sandal	5,2
n03179701	desk	5,2
n02879718	bow	5,2
n02391049	zebra	5,3
n04465501	tractor	5,3
n02802426	basketball	5,3
n04330267	stove	5,3
n03804744	nail	5,4
n02843684	birdhouse	5,4
n01944390	snail	5,5
n11939491	daisy	5,5
n04026417	purse	5,5
n03868863	mask	5,5

ImageNet class	Class Name	AoA-rating
n07802026	hay	5,6
n02870880	bookcase	5,6
n02363005	beaver	5,7
n03742115	chest	5,7
n02906734	broom	5,7
n03481172	hammer	5,7
n04579432	whistle	5,7
n02342885	hamster	5,8
n03000684	chain	5,8
n02119789	ox	5,9
n02443114	pole	5,9
n01945685	slug	5,9
n04507155	umbrella	5,9
n04584207	wig	6
n03125729	cradle	6
n04208210	shovel	6
n03690938	lotion	6,1
n04476259	tray	6,1
n04325704	stole	6,1
n02701002	ambulance	6,2
n04154565	screwdriver	6,2
n07715103	cauliflower	6,2
n03627232	knot	6,2
n03876231	paintbrush	6,2
n04398044	teapot	6,2
n03291819	envelope	6,3
n02007558	flamingo	6,3
n07734744	mushroom	6,4
n03877845	palace	6,5
n04273569	speedboat	6,5
n02730930	apron	6,5
n02088632	tick	6,5
n02229544	cricket	6,5
n02091635	otter	6,6
n04311174	eel	6,6
n12267677	acorn	6,6
n01910747	jellyfish	6,6
n03109150	screw	6,7
n02910353	buckle	6,7
n04442312	toaster	6,7
n09472597	volcano	6,7
n02971356	carton	6,8

ImageNet class	Class Name	AoA-rating	ImageNet class	Class Name	AoA-rating
n03794056	mousetrap	6,8	n03733131	maypole	8,1
n07695742	pretzel	6,8	n02787622	banjo	8,1
n04522168	vase	6,8	n07875152	potpie	8,1
n07565083	menu	6,9	n02699494	altar	8,2
n02326432	hare	6,9	n04004767	printer	8,2
n04296562	stage	6,9	n03841143	dome	8,2
n01833805	hummingbird	7	n02051845	pelican	8,2
n04429376	throne	7	n04525038	velvet	8,2
n07860988	dough	7	n07753113	fig	8,3
n02106030	collie	7,1	n03825788	nipple	8,4
n04540053	volleyball	7,1	n04317175	stethoscope	8,5
n02951358	canoe	7,1	n02410509	bison	8,5
n03388043	fountain	7,1	n03100240	convertible	8,5
n03530642	honeycomb	7,1	n04040759	radiator	8,5
n04009552	projector	7,2	n02437616	llama	8,6
n02749479	rifle	7,2	n03773504	missile	8,6
n02109961	ski	7,2	n02815834	beaker	8,6
n02669723	gown	7,2	n04355338	sundial	8,7
n04270147	spatula	7,3	n01843383	toucan	8,7
n03482405	hamper	7,3	n02666196	abacus	8,7
n03871628	packet	7,4	n02102318	cock	8,7
n01784675	centipede	7,4	n04251144	snorkel	8,7
n04606251	wreck	7,4	n02276258	admiral	8,7
n03141823	crutch	7,4	n04067472	reel	8,7
n03045698	cloak	7,4	n03933933	pier	8,7
n07932039	eggnog	7,5	n03980874	poncho	8,8
n04456115	torch	7,5	n01818515	macaw	8,8
n02978881	cassette	7,5	n03874599	padlock	8,8
n03255030	dumbbell	7,6	n01580077	jay	8,9
n03944341	pinwheel	7,6	n03803284	muzzle	8,9
n02447366	badger	7,7	n04523525	vault	9,1
n03662601	lifeboat	7,7	n04259630	sombrero	9,1
n01774750	tarantula	7,8	n03720891	maraca	9,1
n03250847	drumstick	7,8	n01616318	vulture	9,1
n02950826	cannon	7,8	n03498962	hatchet	9,1
n01770393	scorpion	7,8	n02804414	bassinet	9,2
n02108089	boxer	7,8	n04485082	tripod	9,3
n02012849	crane	7,9	n03785016	moped	9,4
n02012849	crane	7,9	n04336792	stretcher	9,5
n03127925	crate	7,9	n02423022	gazelle	9,6
n02441942	weasel	7,9	n02137549	mongoose	9,6
n03495258	harp	7,9	n03956157	planetarium	9,6
n04423845	thimble	8,1	n04332243	strainer	9,6
n04487394	trombone	8,1	n01806567	quail	9,6
n02486410	baboon	8,1	n03633091	ladle	9,7

ImageNet class	Class Name	AoA-rating	ImageNet class	Class Name	AoA-rating
n03782006	monitor	9,7	n03786901	mortar	11,5
n01592084	chickadee	9,8	n02088364	beagle	11,6
n02454379	armadillo	9,9	n03467068	guillotine	11,8
n01807496	partridge	9,9	n07613480	trifle	11,8
n03527444	holster	10	n02794156	barometer	11,9
n03868242	oxcart	10	n03680355	Loafer	11,9
n03447447	gondola	10,1	n04147183	schooner	12,4
n02113186	cardigan	10,1	n02490219	marmoset	12,5
n09288635	geyser	10,1	n01847000	drake	12,5
n02859443	boathouse	10,2	n04136333	sarong	12,9
n03781244	monastery	10,3	n02091134	whippet	13,1
n04376876	syringe	10,4	n03788195	mosque	13,2
n02804610	bassoon	10,4	n03297495	espresso	13,3
n02442845	mink	10,4	n04532670	viaduct	13,3
n01582220	magpie	10,5	n03837869	obelisk	13,7
n01943899	conch	10,5	n04501370	turnstile	13,7
n03443371	goblet	10,7	n02361337	marmot	13,9
n04277352	spindle	10,7	n02389026	sorrel	14
n01883070	wombat	11,1	n04141327	scabbard	14,1
n02790996	barbell	11,2	n02011460	bittern	14,5
n07768694	pomegranate	11,2	n02981792	catamaran	15,6
n04111531	rotisserie	11,3	n02018795	bustard	15,8
n03617480	kimono	11,4	n03146219	cuirass	15,9
n04612504	yawl	11,4	n02006656	spoonbill	16
n09193705	alp	11,5	n01560419	bulbul	17,2

C Training code

```
import tensorflow as tf
from tensorflow.keras.callbacks import Callback
#from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Dense, Flatten, UpSampling2D,Conv3D,Lambda,Reshape,Permute,Input,Add, ConvLSTM2D,Subtract, UpSampling2D
#from tensorflow.keras.layers import BatchNormalization, TimeDistributed, LeakyReLU, concatenate,ReLU,Multiply,Lambda, GlobalMaxPool2D,LSTM,GRU,SpatialDropout2D, RepeatVector
from tensorflow.keras.layers import Dense,Input
from tensorflow.keras import backend as K
from tensorflow.keras.models import Model
import numpy as np,sys
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import decode_predictions
import os
import random


def Train():

    NoOfClasses = 1000
    Epochs = 29
    Batch_size = 8
    AmountOfImages = 6000
    Val_split = 0.2

    #Prepare the classes to integer number
    ids = {}
    f = open('./ILSVRC2017_CLS-LOC/mappedclasses.txt','r') #mappedclasses.txt can be found in Appendix D
    lines = f.readlines()

    for i in lines:
        temp = i.strip()
        temp = temp.split(' ')
        ids[temp[0]] = int(temp[1])

    #get the baseline resnet model to build layer ontop of
    resnet = tf.keras.applications.ResNet50(weights='imagenet',include_top=False, pooling='avg')

    #Adapt the resnet output as the input for the new layers
    input = Input((224,224,3))
    features = resnet(input)

    #create two Dense layers
    fc1 = Dense(2048,activation='relu')(features)
    fc2 = Dense(NoOfClasses,activation='softmax')(fc1)

    #Create the model with the extra layers
    AoA_model = Model(inputs=input,outputs=fc2)
    AoA_model.compile(optimizer = tf.keras.optimizers.Adam(amsgrad=True), loss = 'sparse_categorical_crossentropy',metrics=['accuracy'])


```

```
    #initialize empty arrays to fill with images and labels for training
    TrainImages = []
    TrainLabels = []

    #Create or load an array with the classes to train on (e.g.: TrainCats = ['n03782006','n01592084','n02454379','n01807496','n03527444','n03868242','n03447447','n02113186','n09288635','n02859443','n03781244','n04376876','n02804610','n02442845','n01582220','n01943899','n03443371','n04277352','n01883070','n02790996'] )
    TrainCats = []

    for i in range(1,AmountOfImages+1):
        #Load a random image from any of the classes in TrainCats
        folder = random.choice(TrainCats)
        file = random.choice(os.listdir('./ILSVRC2017_CLS-LOC/train/'+folder+'/'))
        img = image.load_img('./ILSVRC2017_CLS-LOC/train/'+folder+'/'+file, target_size=(224, 224,3))

        #Load the image to an array to preprocess it for ResNet50
        img = image.img_to_array(img)
        img = tf.keras.applications.resnet.preprocess_input(img)
        img = np.expand_dims(img,0)

        #Append the image and the appropriate label to the trainin arrays
        TrainImages.append(img)
        TrainLabels.append(ids[folder])

    #Stack the images vertically to make the keras.fit() recognise them
    TrainImages = np.vstack(TrainImages)

    #Reshape the labels array in the same way
    TrainLabels = np.array(TrainLabels)
    TrainLabels= np.reshape(TrainLabels,(6000,1))

    #Create a checkpoint system that saves the model and it's weights, based on the best validation accuracy
    checkpoint = tf.keras.callbacks.ModelCheckpoint("best_model_29.hdf5", monitor='val_accuracy', verbose=1,
        save_best_only=True, mode='auto', period=1,histogram_freq=0)

    #Actually train the model
    AoA_model.fit(x=TrainImages,y=TrainLabels,batch_size=Batch_size,epochs=Epochs,validation_split=Val_split,callbacks=[checkpoint])
```

Listing 1: Training program

D Mappedclasses.txt

Mappedclasses.txt:

n02119789 278
n02100735 212
n02110185 250
n02096294 193
n02102040 217
n02066245 147
n02509815 387
n02124075 285
n02417914 350
n02123394 283
n02125311 286
n02423022 176
n02346627 334
n02077923 150
n02110063 249
n02447366 362
n02109047 246
n02089867 166
n02102177 218
n02091134 172
n02092002 177
n02071294 148
n02442845 357
n02504458 386
n02092339 178
n02098105 202
n02096437 194
n02114712 271
n02105641 229
n02128925 290
n02091635 175
n02088466 163
n02096051 191
n02117135 275
n02138441 299
n02097130 197
n02493509 380
n02457408 364
n02389026 339
n02443484 359
n02110341 251
n02089078 165
n02086910 157
n02445715 361
n02093256 179
n02113978 268
n02106382 233
n02441942 356
n02113712 266
n02113186 264
n02105162 225
n02415577 349
n02356798 335
n02488702 375
n02123159 282
n02098413 204
n02422699 352
n02114855 272
n02094433 187
n02111277 256
n02132136 294
n02119022 188
n02091467 174
n02106550 234
n02422106 351
n02091831 176
n02120505 280
n02104365 223
n02086079 154
n02112706 262
n02098286 203
n02095889 190
n02484975 370
n02137549 298
n02500267 384
n02129604 3
n02090721 170
n02396427 342
n02108000 241
n02391049 340
n02412080 5
n02108915 245
n02480495 365
n02110806 253
n02128385 288
n02107683 239
n02085936 153
n02094114 185
n02087046 158
n02100583 211
n02096177 192
n02494079 382
n02105056 224
n02101556 216
n02123597 284
n02481823 367
n02105505 228
n02088094 160
n02085782 152
n02489166 376
n02364673 338
n02114548 270
n02134084 296
n02480855 366
n02090622 169
n02113624 265
n02093859 183
n02403003 37
n02097298 199
n02108551 244
n02493793 381
n02107142 236
n02096585 195
n02107574 238
n02107908 240
n02086240 155
n02102973 221
n02112018 259
n02093647 181
n02397096 343
n02437312 354
n02483708 369
n02097047 196
n02106030 231
n02099601 207
n02093991 184
n02110627 252
n02106166 232
n02326432 331
n02108089 242
n02097658 201
n02088364 162
n02111129 255
n02100236 210
n02486261 371
n02115913 274
n02486410 372
n02487347 373
n02099849 209
n02108422 243
n02104029 222
n02492035 378
n02110958 254
n02099429 206
n02094258 186
n02099267 205
n02395406 54
n02112350 261
n02109961 248
n02101388 215
n02113799 267
n02095570 189
n02128757 289
n02101006 214
n02115641 273
n02097209 198
n02342885 333
n02097474 200
n02120079 279
n02095314 188
n02088238 161
n02408429 346
n02133161 295
n02328150 332
n02410509 347
n02492660 379
n02398521 344
n02112137 260
n02510455 388
n02093428 180
n02105855 230
n02111500 257
n02085620 151
n02123045 281
n02490219 377
n02099712 208
n02109525 247
n02454379 363
n02111889 258
n02088632 164
n02090379 168
n02443114 358
n02361337 336
n02105412 227
n02483362 368
n02437616 355
n02107312 237
n02325366 330
n02091032 171
n02129165 150
n02102318 219
n02100877 213
n02074367 149
n02504013 385
n02363005 337
n02102480 220
n02113023 263
n02086646 156
n02497673 383
n02087394 159
n02127052 287
n02116738 275
n02488291 374
n02091244 173
n02114367 269
n02130308 293
n02089973 167
n02105251 226
n02134418 297
n02093754 182

n02106662 235
n02444819 175
n01882714 105
n01871265 101
n01872401 102
n01877812 104
n01873310 103
n01883070 106
n04086273 763
n04507155 879
n04147183 780
n04254680 805
n02672831 401
n02219486 48
n02317335 327
n01968897 117
n03452741 579
n03642806 620
n07745940 949
n02690373 404
n04552348 895
n02692877 405
n02782093 417
n04266014 812
n03344393 554
n03447447 576
n04273569 814
n03662601 625
n02951358 472
n04612504 914
n02981792 484
n04483307 871
n03095699 510
n03673027 404
n03947888 724
n02687172 403
n04347754 833
n04606251 864
n03478589 586
n04389033 639
n03773504 657
n02860847 450
n03218198 537
n02835271 444
n03792782 671
n03393912 565
n03895866 705
n02797295 428
n04204347 791
n03791053 670
n03384352 561
n03272562 547
n04310018 820
n02704792 408
n02701002 407
n02814533 436
n02930766 324
n03100240 511
n03594945 609
n03670208 627
n03770679 656
n03777568 661
n04037443 751
n04285008 817
n03444034 573
n03445924 575
n03785016 665
n04252225 803
n03345487 555
n03417042 569
n03930630 717
n04461696 864
n04467665 867
n03796401 675
n03977966 734
n04065272 757
n04335435 829
n04252077 802
n04465501 866
n03776460 660
n04482393 870
n04509417 880
n03538406 603
n03599486 612
n03868242 690
n02804414 431
n03125729 116
n03131574 520
n03388549 564
n02870880 453
n03018349 495
n03742115 648
n03016953 493
n04380533 846
n03337140 553
n03891251 703
n02791124 423
n04429376 857
n03376595 559
n04099969 765
n04344873 831
n04447861 861
n03179701 526
n03982430 736
n03201208 532
n03290653 548
n04550184 894
n07742313 948
n07747607 950
n07749582 951
n07753113 952
n07753275 953
n07753592 954
n07754684 955
n07760859 956
n07768694 957
n12267677 941
n12620546 41
n13133613 105
n11879895 984
n12144580 66
n12768682 990
n03854065 593
n04515003 881
n03017168 494
n03249569 541
n03447721 149
n03720891 641
n03721384 642
n04311174 822
n02787622 420
n02992211 486
n04536866 889
n03495258 593
n02676566 402
n03272010 546
n03110669 513
n03394916 566
n04487394 875
n03494278 593
n03840681 684
n03884397 699
n02804610 432
n03838899 683
n04141076 776
n03372029 558
n11939491 985
n12057211 986
n09246464 500
n09468604 979
n09193705 140
n09472597 980
n09399592 976
n09421951 977
n09256479 973
n09332890 975
n09428293 978
n09288635 974
n03498962 596
n03041632 499
n03658185 623
n03954731 726
n03995372 740
n03649909 621
n03481172 4
n03109150 512
n02951585 473
n03970156 731
n04154565 784
n04208210 792
n03967562 730
n03000684 491
n01514668 7
n01514859 8
n01518878 9
n01530575 10
n01531178 11
n01532829 12
n01534433 13
n01537544 14
n01558993 15
n01560419 16
n01580077 17
n01582220 18
n01592084 19
n01601694 20
n01608432 21
n01614925 22
n01616318 23
n01622779 24
n01795545 80
n01796340 81
n01797886 82
n01798484 83
n01806143 84
n01806567 85
n01807496 82
n01817953 87
n01818515 88
n01819313 89
n01820546 90
n01824575 91
n01828970 92
n01829413 93
n01833805 94
n01843065 95
n01843383 96
n01847000 97
n01855032 98
n01855672 99
n01860187 100
n02002556 127
n02002724 128
n02006656 129
n02007558 130
n02009912 132
n02009229 131
n02011460 133
n02012849 134
n02013706 135
n02018207 137
n02018795 138
n02025239 139
n02027492 140
n02028035 141
n02033041 142
n02037110 143
n02017213 136
n02051845 144
n02056570 145

n02058221 146
n01484850 2
n01491361 3
n01494475 4
n01496331 5
n01498041 6
n02514041 389
n02536864 391
n01440764 0
n01443537 1
n02526121 390
n02606052 392
n02607072 393
n02643566 396
n02655020 397
n02640242 394
n02641379 26
n01664065 33
n01665541 34
n01667114 35
n01667778 36
n01669191 37
n01675722 38
n01677366 39
n01682714 40
n01685808 41
n01687978 42
n01688243 43
n01689811 44
n01692333 45
n01693334 46
n01694178 47
n01695060 48
n01704323 51
n01697457 49
n01698640 50
n01728572 52
n01728920 53
n01729322 54
n01729977 55
n01734418 56
n01735189 57
n01737021 58
n01739381 59
n01740131 60
n01742172 61
n01744401 62
n01748264 63
n01749939 64
n01751748 65
n01753488 66
n01755581 67
n01756291 68
n01629819 25
n01630670 26
n01631663 27
n01632458 28
n01632777 29
n01641577 30
n01644373 31
n01644900 32
n04579432 902
n04592741 318
n03876231 696
n03483316 589
n03868863 691
n04251144 801
n03691459 632
n03759954 650
n04152593 556
n03793489 673
n03271574 545
n03843555 686
n04332243 828
n04265275 811
n04330267 827
n03467068 583
n02794156 426
n04118776 131
n03841143 685
n04141975 778
n02708093 409
n03196217 530
n04548280 892
n03544143 604
n04355338 835
n03891332 704
n04328186 826
n03197337 531
n04317175 823
n04376876 845
n03706229 635
n02841315 447
n04009552 745
n04356056 837
n03692522 633
n04044716 755
n02879718 456
n02950826 471
n02749479 413
n04090263 413
n04008634 744
n03085013 508
n04505470 878
n03126707 134
n03666591 626
n02666196 398
n02977058 480
n04238763 798
n03180011 527
n03485407 590
n03832673 681
n06359193 916
n03496892 595
n04428191 856
n04004767 742
n04243546 297
n04525305 886
n04179913 786
n03602883 613
n04372370 844
n03532672 600
n02974003 479
n03874293 694
n03944341 723
n03992509 739
n03425413 571
n02966193 476
n04371774 843
n04067472 758
n04040759 581
n04019541 746
n03492542 592
n04355933 836
n03929660 714
n02965783 475
n04258138 807
n04074963 761
n03208938 535
n02910353 464
n03476684 584
n03627232 616
n03075370 507
n03874599 695
n03804744 113
n04127249 772
n04153751 512
n03803284 676
n04162706 785
n04228054 248
n02948072 470
n03590841 607
n04286575 818
n04456115 862
n03814639 678
n03933933 713
n04485082 872
n03733131 645
n03794056 674
n04275548 815
n01768244 69
n01770081 70
n01770393 71
n01773157 72
n01773549 73
n01773797 72
n01774384 75
n01774750 76
n01775062 77
n01776313 78
n01784675 79
n01990800 126
n01978287 118
n01978455 119
n01980166 120
n01981276 121
n01983481 122
n01984695 123
n01985128 123
n01986214 125
n02165105 300
n02165456 301
n02167151 302
n02168699 303
n02169497 304
n02172182 305
n02174001 306
n02177972 307
n02190166 308
n02206856 92
n02226429 311
n02229544 312
n02231487 313
n02233338 314
n02236044 315
n02256656 316
n02259212 317
n02264363 318
n02268443 319
n02268853 320
n02276258 321
n02277742 322
n02279972 323
n02280649 324
n02281406 325
n02281787 326
n01910747 107
n01914609 108
n01917289 109
n01924916 110
n01930112 111
n01943899 112
n01944390 113
n01945685 114
n01950731 115
n01955084 116
n02319095 328
n02321529 329
n03584829 606
n03297495 550
n03761084 651
n03259280 544
n04111531 766
n04442312 859
n04542943 891
n04517823 882
n03207941 534
n04070727 760
n04554684 534
n03133878 521

n03400231 567
n04596742 909
n02939185 469
n03063689 505
n04398044 849
n04270147 813
n02699494 406
n04486054 873
n03899768 706
n04311004 821
n04366367 839
n04532670 888
n02793495 73
n03457902 580
n03877845 498
n03781244 663
n03661043 624
n02727426 410
n02859443 449
n03028079 497
n03788195 668
n04346328 832
n03956157 727
n04081281 762
n03032252 498
n03529860 598
n03697007 634
n03065424 506
n03837869 682
n04458633 863
n02980441 483
n04005630 743
n03461385 582
n02776631 415
n02791270 424
n02871525 454
n02927161 467
n03089624 509
n04200800 788
n04443257 860
n04462240 865
n03388043 562
n03042490 500
n04613696 915
n03216828 536
n02892201 458
n03743016 649
n02788148 421
n02894605 460
n03160309 67
n03000134 489
n03930313 716
n04604644 912
n04326547 825
n03459775 581
n04239074 799
n04501370 877
n03792972 672
n04149813 781
n03530642 599
n03961711 729
n03903868 708
n02814860 437
n07711569 935
n07720875 945
n07714571 936
n07714990 937
n07715103 938
n07716358 939
n07716906 940
n07717410 941
n07717556 942
n07718472 329
n07718747 944
n07730033 946
n07734744 947
n04209239 794
n03594734 608
n02971356 478
n03485794 591
n04133789 774
n02747177 412
n04125021 771
n07579787 461
n03814906 679
n03134739 522
n03404251 568
n04423845 855
n03877472 697
n04120489 770
n03062245 503
n03014705 492
n03717622 640
n03777754 662
n04493381 435
n04476259 868
n02777292 416
n07693725 931
n03998194 741
n03617480 614
n07590611 926
n04579145 901
n03623198 615
n07248320 921
n04277352 816
n04229816 796
n02823428 440
n03127747 518
n02877765 455
n04435653 858
n03724870 570
n03710637 638
n03920288 712
n03379051 560
n02807133 433
n04399382 850
n03527444 597
n03983396 737
n03924679 713
n04532106 887
n06785654 918
n03445777 574
n07613480 927
n04350905 639
n04562935 900
n03325584 552
n03045698 501
n07892512 966
n03250847 542
n04192698 787
n03026506 496
n03534580 601
n07565083 922
n04296562 819
n02869837 452
n07871810 962
n02799071 429
n03314780 551
n04141327 777
n04357314 838
n02823750 441
n13052670 996
n07583066 924
n03637318 619
n04599235 911
n07802026 958
n02883205 457
n03709823 636
n04560804 899
n02909870 463
n03207743 533
n04263257 809
n07932039 969
n03786901 666
n04479046 869
n03873416 693
n02999410 488
n04367480 840
n03775546 659
n07875152 964
n04591713 907
n04201297 789
n02916936 465
n03240683 540
n02840245 446
n02963159 264
n04370456 841
n03991062 28
n02843684 448
n03482405 588
n03942813 722
n03908618 709
n03902125 707
n07584110 925
n02730930 411
n04023962 747
n02769748 414
n10148035 982
n02817516 439
n03908714 710
n02906734 462
n03788365 669
n02667093 399
n03787032 667
n03980874 735
n03141823 523
n03976467 732
n04264628 810
n07930864 334
n04039381 752
n06874185 920
n04033901 749
n04041544 754
n07860988 961
n03146219 524
n03763968 652
n03676483 629
n04209133 793
n03782006 664
n03857828 688
n03775071 658
n02892767 459
n07684084 930
n04522168 883
n03764736 653
n04118538 768
n03887697 700
n13044778 995
n03291819 549
n03770439 655
n03124170 515
n04487081 874
n03916031 711
n02808440 435
n07697537 934
n12985857 991
n02917067 466
n03938244 721
n15075141 999
n02978881 481
n02966687 477
n03633091 618
n13040303 994
n03690938 631
n03476991 585
n02669723 400
n03220513 538
n03127925 519
n04584207 903
n07880968 965

n03937543 720	n03871628 692	n02834397 443	n03595614 610	n13054560 997
n03000247 490	n03733281 646	n03888257 701	n04146614 779	n10565667 983
n04418357 854	n03976657 358	n04235860 797	n03598930 611	n03950228 725
n04590129 905	n03535780 602	n04404412 851	n03958227 728	n03729826 644
n02795169 412	n04259630 808	n04371430 842	n04069434 759	n02837789 445
n04553703 896	n03929855 715	n03733805 647	n03188531 529	n04254777 806
n02783161 418	n04049303 756	n07920052 550	n02786058 419	n02988304 485
n02802426 430	n04548362 893	n07873807 963	n07615774 929	n03657121 622
n02808304 434	n02979186 482	n02895154 461	n04525038 885	n04417672 853
n03124043 514	n06596364 917	n04204238 790	n04409515 852	n04523525 884
n03450230 400	n03935335 719	n04597913 910	n03424325 570	n02815834 438
n04589890 904	n06794110 919	n04131690 773	n03223299 539	n09229709 971
n12998815 992	n02825657 442	n07836838 960	n03680355 630	n07697313 933
n02992529 487	n03388183 563	n09835506 981	n07614500 928	n03888605 702
n03825788 680	n04591157 906	n03443371 572	n07695742 932	n03355925 557
n02790996 422	n04540053 890	n13037406 993	n04033995 750	n03063599 504
n03710193 637	n03866082 689	n04336792 830	n03710721 638	n04116512 767
n03630383 617	n04136333 775	n04557648 898	n04392985 848	n04325704 824
n03347037 556	n04026417 748	n03187595 528	n03047690 502	n07831146 959
n03769881 654	n02865351 451	n04254120 804	n03584254 605	n03255030 543

E Validation code

```
import os
import sys
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import tensorflow_datasets as tfds
from keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import decode_predictions
import keras.applications
import random
from xml.dom import minidom
from tqdm import tqdm
import cv2

#Load or create an array with each class in ImageNet (e.g.: imagenet = ['n02119789', 'n02100735', 'n02110185', 'n02096294',....])
imagenet = []

#Create an empty dictionary to create a map of all classes and tha accuracy of the validation images
classdict = {}
#insert the classes with an empty array
for k in imagenet:
    array = []
    classdict.update({k:array})

#load array from file with all the correct classes mapped to each image (e.g. Image ILSVRC2012_val_00000001 correspond to class n01751748, so the first line in the file is n01751748)
labels = []
f = open('./ILSVRC2017_CLS-LOC/val/ILSVRC2012_validation_ground_truth.txt','r')
lines = f.readlines()
for i in lines:
    labels.append(i.strip())
labels = np.array(labels)

#load the model to validate
ValModel = tf.keras.models.load_model('ModelToLoad.hdf5')

#iterate over each image (50k validation images)
for i in range(1,50001):

    #load image and preprocess
    fname = './ILSVRC2017_CLS-LOC/val/ILSVRC2012_val_'+ str(i).zfill(8)+'.JPEG'
    img = image.load_img(fname, target_size=(224, 224,3))
    img = image.img_to_array(img)
    img = tf.keras.applications.resnet.preprocess_input(img)
    img = np.expand_dims(img,0)

    #Use the ValModel to predict the class
    pred = ValModel.predict(img)
    decodedPred = np.squeeze(decode_predictions(pred, top=1))  ### decodedPred = [id,name,prob]

    #check if prediciton is correct
```

```
    correct = 0
    if decodedPred[0] == labels[i-1]:
        correct = 1

    #get the correct class' array
    arr = classdict.get(labels[i-1])
    #add whether the prediction was correct (1) or incorrect (0)
    arr.append(correct)
    #re-enter the new array to that class
    classdict.update({labels[i-1]:arr})

    #This results in a dictionary with 50.000 entries with their own array
    containing 50 ones and zeros

#Create a new dictionary to house each class and it's accuracy
accdict = {}

for k in imagenet:
    CorrectArray = classdict.get(k)
    TotalPredictions = len(CorrectArray)
    #Filter all the 0 values out of the array
    CorrectArray = [i for i in CorrectArray if i != 0]
    #This gives a percentage of how many images were predicted accurately
    accdict.update({k:((len(suc)/totaal)*100)})

#Write this dictionary to a file to save for results
file = open('AccuracyPerClass.txt',"w")
str_dictionary = repr(accdict)
file.write(str_dictionary)
file.close
```

Listing 2: Validation program