Distillation and Partial Model Freezing for Continual Learning in Neural Fields

Wouter Visser

***Abstract*—Neural fields have increasingly been utilised as signal representations. Using neural networks, they can be applied as solutions to problems such as creating continuous representations and solving inverse problems.**

In some use cases, the signals represented are not only spatial but also temporal. In practice, when such a signal has to be represented, it can be beneficial to incorporate measurements into an existing neural field as soon as they are available. However, this presents a problem, as neural networks, which neural fields are based on, can forget knowledge learnt when being presented with new knowledge.

Considerable research has been conducted on remedying this problem in general neural networks. However, differences in model input and architecture compared to general neural networks mean that the results from this research may not be directly applicable to neural fields.

In this work, we investigate how continual learning affects neural fields of different architectures. Additionally, we demonstrate two methods that can be used to remedy the problems caused by continual learning in neural fields. First, we demonstrate how the model trained on earlier tasks can be reused to prevent performance degradation through knowledge distillation when learning a subsequent task. Second, we demonstrate how, for models using a DINER architecture, a part of the model can be frozen, thereby lowering performance loss while still allowing the model to learn to represent new measurements.

In summary, we show that the continual learning of neural fields of various architectures is, to a reasonable extent, feasible.

I. Introduction

In recent years, neural fields, also known as implicit neural representations, have emerged as a novel method for continuously representing signals on low-dimensional domains [1]. In neural fields, a signal is modelled with the help of a multi-layer perceptron (MLP). Given a function on a spatial or spatiotemporal domain, a neural field takes coordinates in space and/or time as input and is trained to output the signal or signals at that coordinate. This training is often done using reference outputs for some coordinates. During inference, the value can be queried for any coordinate in the coordinate space. This means the training creates a continuous representation based on the training data.

A neural field can be trained directly, but training of neural fields can also be achieved indirectly through a differentiable transformation of the output. Differentiability is necessary so the gradients can be backpropagated through the transformation to the parameters of the neural field. The ability to create an interpolatable representation from measurements from inverse problems makes neural fields an attractive solution for signal representation. Because of this, neural fields have seen use in diverse applications like 3D shape representation [2], medical image reconstruction [3]–[5] and novel view synthesis [6], [7].

These applications utilise spatial coordinates, but neural fields can also be used to represent spatio-temporal signals. This has been used for areas like video representation [8] and disease monitoring [9]. Most neural field methods assume that all datapoints used for fitting are available simultaneously, allowing for a single fitting stage. In practice, this might not always be the case. For instance, if the time between measurements is long, a trained neural field might already be needed between measurements. In this case, it would be advantageous to process the available signals as they become available. However, if the neural field has to be changed when additional measurements become available, it would be inefficient to retrain it completely. Additionally, the measurements used to train the model initially may no longer be available. Instead, fine-tuning the existing parameters to fit the new measurements would be more efficient. This could present a problem, as MLPs suffer from a problem called catastrophic forgetting [10], meaning that when training data is incrementally introduced to an MLP, data learnt first tends to be forgotten.

A large amount of research has been done into remedying the effects of catastrophic forgetting, i.e. 'continual learning' [11]. Most of the previously done research has been on more general machine learning applications, such as image classification. Some differences between these applications and neural fields mean that some of the research might not fully transfer.

First, there is a significant difference in the input. Instead of the high-dimensional and complex input used in many general machine learning tasks, neural fields have a very structured and low-dimensional input. Theoretical research on continual learning through the Neural Tangent Kernel shows that the similarity of the data contributes to forgetting [12]. Due to the high similarity of the input for neural fields, they may be especially prone to catastrophic forgetting.

Additionally, neural fields often have a distinct architecture compared to other neural network applications. This is because the signals learnt by neural fields often have a high frequency. This is a problem for neural networks, as they are slower at learning higher-frequency signals, known as the spectral bias of neural networks [13]. To solve this problem, changes are made to the models to make them more resistant to this, often by changing the activation function of the network [14]–[16]. The chosen activation functions also affect how a neural network behaves in a continual learning scenario [17].

In this work, we analyse how commonly used models for neural fields and their hyperparameters affect continual learn-

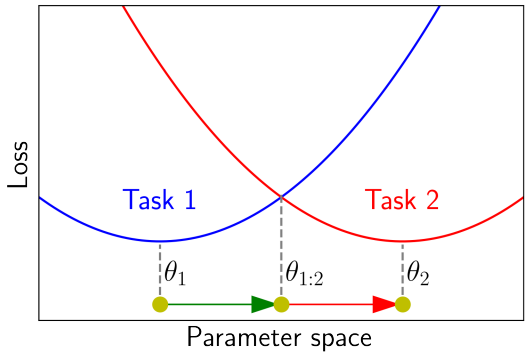


Fig. 1: The stability-plasticity tradeoff visualised. A model should have the plasticity to move away from θ_1, but the stability not to move too far to θ_2.

ing behaviour. Next, we show two methods that can be used to remedy catastrophic forgetting. First, the structure of the input can be used to allow an old version of the model to teach a new version on old tasks through knowledge distillation. Second, the architecture employed by hashmap-based models enables the separation of the output representation from the input coordinate structure. This means we can reduce forgetting by learning only one of these parts while preventing the other part from learning. Through experiments with representations of natural and medical images, we show how these can contribute towards continual learning in neural fields.

II. Background

The big challenge for continual learning is *catastrophic forgetting* [10], [17]. Catastrophic forgetting is the problem where, when a machine learning model is incrementally trained on tasks, it loses performance on early tasks [18], [19]. If a model or method can overcome forgetting, it is described as stable.

However, this is only one aspect of continual learning. Not only can the model forget old knowledge, but it can also lose the ability to learn new knowledge. This aspect is known as plasticity, where a model that can learn new knowledge is described as plastic. In the field of continual learning, most research has focused on overcoming catastrophic forgetting; however, some studies indicate that models are also unable to acquire new knowledge infinitely [20].

Together, this problem is called the stability-plasticity tradeoff. Machine learning models should remain plastic enough to learn new knowledge, but stable enough to remember what is learnt from earlier samples. This principle is visualised in Figure 1. Here, θ_1 denotes the model's parameters after training on task 1. Plasticity means the ability to move the parameters away from θ_1, ideally to $\theta_{1:2}$, the parameters with the best performance on tasks 1 and 2. Stability is the ability to resist continuing this movement to θ_2, the ideal parameters for just task 2.

Several techniques have been developed for continual learning. In general, these can be divided into five categories.

a) Weight Regularisation: The first group of methods restrict the weight changes between tasks. Starting from the second task, these methods add a regularisation term to the loss function of the task learnt. This regularisation term consists of the difference between the model parameters after training a previous task and the current parameters. However, as some parameters are more important to the model's output on a task than others, this difference is weighted by an importance metric for each parameter. Examples of this include the Fisher information for Elastic Weight Consolidation (EWC) [18] or the history of gradients during training for Synaptic Intelligence [21].

b) Experience Replay: Another method used is selectively saving samples from previous tasks. Then, during the training of the next task, these samples are replayed to the model, so it does not forget them. Research surrounding this method focuses on two key questions: Which samples are the most effective to save (e.g., [22]), and which sample is the most suitable to use during training for the new task (e.g., [23]).

c) Knowledge Distillation: A downside of saving previous samples is that it requires storage, as well as prior knowledge that continual learning will occur. To solve this problem, a previous version of the model can be used. After training on a task, the model should perform well on that task. So, when trying to maintain performance, a previous version of the model can be used instead of saving the solution to these inputs. Then, for some input, the currently training model is also trained to maintain the output of the previous model. This method is known as knowledge distillation. However, determining the input to use for distillation is a challenge, for which previous works have proposed using training data from previous tasks [24], unlabeled data [25], or new training data [26].

d) Optimisation Strategies: Fourth, the optimisation strategy can also be adapted to prevent forgetting. An example of this is Orthogonal Gradient Descent [27]. In this method, the derivative of the loss on a task with respect to the parameters is stored after training each task. Then, during the training of subsequent tasks, the calculated weight updates are changed such that they are orthogonal to the stored derivative.

e) Architecture Changes: Lastly, the model itself can also be adapted to prevent forgetting. This means designating certain parts of the model to different tasks. One possibility is to use parameters that are important for a specific task selectively through a learnt binary mask (e.g. [28]). Another option is creating specific parts of the model for different tasks (e.g. [29]).

III. Related Work

Some works exist that explore continual learning in neural fields. Most of these methods employ a version of knowledge distillation to achieve their goal.

Continual Neural Mapping [30] continuously learns a signed distance function (SDF). This is done by incrementally using measurements from different depth camera views. To prevent forgetting, a combination of knowledge distillation and replay

TABLE I: NeRF continual learning methods and their distillation data generation method

Paper	Ray generation method
Po et al. [31]	Saving previous rays
CLNeRF [32]	Saving centre rays and camera parameters
MEIL-NeRF [33]	Generator neural network
UNIKD [34]	Uncertainty-filtered randomly generated rays

is used. For zero level-set samples, a fixed-size buffer of samples from previous tasks is kept. The previous network is only used for off-surface samples.

A large number of methods [31]–[34] are defined on Neural Radiance Fields (NeRF), a famous use case of neural fields. In NeRF [6], a 3D representation of a scene is constructed using images of that scene. For each pixel of these images, a ray is marched through the neural field, which regularly evaluates the neural field using the position and viewing angle. The neural field returns the colour and density at these positions, which are combined to give an output colour. The loss is calculated on the output colour and the actual pixel value.

In these NeRF continual learning methods, previous rays are either saved or used to generate new ones, depending on the technique. The exact methods used to generate rays per method are shown in Table I. These rays are then marched through the old neural field. The output is then used to supervise the new model.

These methods do not cover all cases for multiple reasons. First, instead of expanding the input domain, these methods focus on specifying information in previously trained regions. Often, the scene represented using these methods remains in the same region of space, but views are incorporated from different angles to correct any mistakes. However, new measurements can also be incorporated in different regions of the input space for which no measurements are available yet. As this is a different approach, findings from these methods do not necessarily generalise to new input regions or outputs that are trained.

Second, these works propose methods that rely on their specific use case to function. For the NeRF models, information about the rays marched through the model in previous tasks is used to prevent forgetting. Similarly, Continual Neural Mapping utilises the structure of SDF to determine whether distillation or replay should be employed and which samples to use. However, when not using these use cases, these methods are not defined.

Lastly, these works each only investigate how to solve the problem for one model architecture, most commonly a positionally encoded ReLU model. This model is often used for the NeRF use case. However, as explained in the introduction, the choice of model architecture likely influences the forgetting behaviour of neural networks. Additionally, as more models are commonly used in neural fields, understanding how these influence forgetting behaviour is important.

IV. Problem Setting

In neural fields, we are interested in approximating a signal consisting of values for every coordinate in a coordinate space.

(a) Full Sample (b) Input Expansion (c) Output Expansion

Fig. 2: Examples of the signal expansion types. Each image shows an example of the first task. Figure 2b shows the top quarter of the image. Subsequent tasks would consist of the other parts of the image. Figure 2c shows only the red band of an RGB image. The following tasks could be the green and blue bands of the image.

This coordinate space can be spatial or spatiotemporal. The representation is done by training a neural network, with parameters θ. Typically, this network is trained using a single training set, where the loss, called $\mathcal{L}_{fit}$, is minimised. This training set typically consists of coordinates for which the corresponding values are known. However, it can also consist of transformations, for which coordinates can be calculated, whose values can be combined to train the neural network. An example of this is images and their associated camera parameters, as seen in novel view synthesis.

In continual learning of neural fields, the training set is split into tasks. Each task has its own associated training set of samples.

A. Signal Expansion Types

We identify two ways in which newly acquired measurements can be integrated into a trained neural field. Figure 2 visualizes an examples of each type for a 2D image.

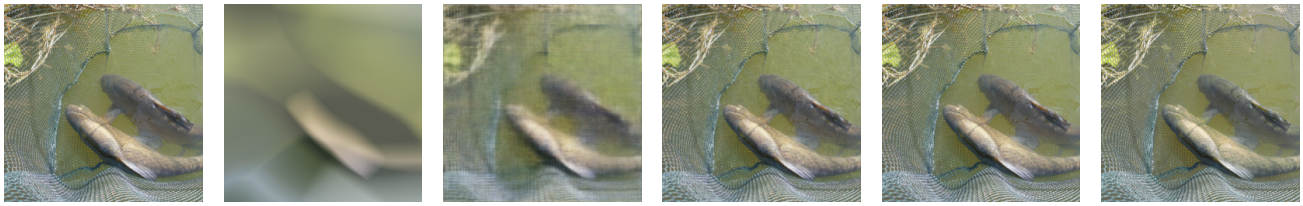
First, data can be acquired from new regions in the spatial or spatio-temporal coordinate space. This means that a model could be trained on measured data from one region and expanded with data from another area, while preserving the information from the previous region. Each task would then consist of measurements from distinct regions in the input space. For instance, measurements can be made at different points in time, which are added to a spatiotemporal neural field when they become available. Alternatively, separate measurements can be made in different areas of space and combined into a unifying representation.

Second, new information may become available for pre-existing regions in the input space. So, for every point in the spatiotemporal coordinate space, the number of outputs increases. The coordinates from each task would then be from the same region. However, during each task, only a subset of output values is available. For example, measurements of different properties in the same domain can be combined into a unifying representation.

V. Methods

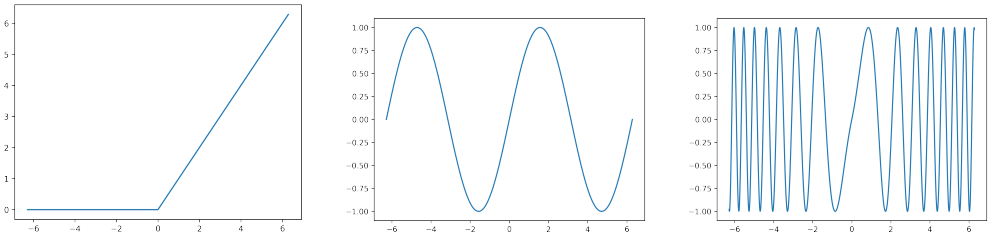
A. Neural Field Architectures

Several models have been proposed for neural fields. The primary difference between the models is how they deal with



(a) Ground Truth (b) ReLU (c) Postional Encoding (d) SIREN (e) FINER (f) DINER

Fig. 3: Comparison of how well each neural network architecture used fits a sample image. The ReLU model does not fit the sample well due to the spectral bias.



(a) ReLU (b) SIREN (c) FINER

Fig. 4: Comparison of the activation functions used.

spectral bias [13]. This is the problem that neural networks are slower to learn high-frequency signals compared to low-frequency ones. An example of this is shown in Figure 3b, where the model fails to accurately fit a sample image. As neural fields often need to represent high-frequency signals, such as images, they frequently suffer from spectral bias. Various techniques have been proposed to address this issue. The different approaches we use are introduced in this section. Examples of how each model fits an image are shown in Figure 3.

1) Postional Encodings: A commonly used method to address the spectral bias problem is to utilise positional encodings. These transform the coordinates into a higher-dimensional space. The encoded coordinates are used as input for a regular ReLU model. We refer to the combined system as a P.E. ReLU model.

The positional encodings are similar to and inspired by the positional encodings used by transformers [35]. We use the positional encodings used by Neural Radiance Fields (NeRF) [6]. This encoding consists of L increasingly high-frequency sine and cosine functions to encode each input variable x, according to the following formula:

$$\begin{aligned}\gamma(x) = [&sin(2^0\pi x), cos(2^0\pi x),\\ &sin(2^1\pi x), cos(2^1\pi x), \dots,\\ &sin(2^{L-2}\pi x), cos(2^{L-2}\pi x)\\ &sin(2^{L-1}\pi x), cos(2^{L-1}\pi x)]\end{aligned}$$

The positional encodings for each input variable are concatenated to form the model input.

2) SIREN: Sinusoidal representation networks (SIREN) [14] replace the activation function of the neural network with a sine function and scale the input to this activation function by a factor called ω_0. This makes the activation function used $\alpha(z) = sin(\omega_0 z)$. The activation function is shown in Figure 4b. The scaling factor allows the activation function to be tuned to suit the frequency of the signal it attempts to fit, with a higher ω_0 enabling the model to fit higher-frequency features.

In the body of the original SIREN paper [14], the authors state that ω_0 should be applied in the first layer of the neural network. However, in the appendix, they state that performance increases if a scaling factor is included in all layers. To differentiate the scaling factor used in the initial layers from the factor in the later layers, we refer to the former as ω_0 and the latter as ω_i

3) FINER: While SIREN and positional encodings enable the fitting of high-frequency features, the (co)sine functions mean the model is only well-suited to a subset of frequencies. To combat this, Flexible spectral-bias tuning in Implicit Neural Representation (FINER) [15] further adjusts the SIREN activation function, creating $\alpha(z) = sin(\omega_i(|z| + 1)z)$. This means that as $|z|$ increases, the frequency of $\alpha(z)$ increases. This is also made visible in Figure 4c.

To further utilise the frequency range, the initialisation scheme is modified, such that the biases are drawn from $\mathcal{U}(-k, k)$, where k is relatively large. The broader range of biases this initialisation scheme creates means that z will also be in a larger range. And as the magnitude of z affects how high-frequency $\alpha(z)$ is, $\alpha(z)$ will have a broader range of frequencies at initialisation.

4) DINER: Instead of a handcrafted function to reduce the relative frequency of the output, like in positional encoding, Disorder-Invariant Implicit Neural Representation (DINER) [36] learns this transformation. This is achieved using a hashmap that maps the input coordinates to an L-dimensional point in a latent space. These new L-dimensional points are then used as input to an MLP. The output values of the hashmap are optimised together with the parameters of the

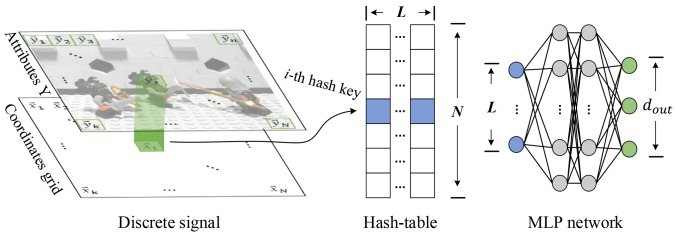


Fig. 5: Architecture used by the DINER models.

Source: [36]

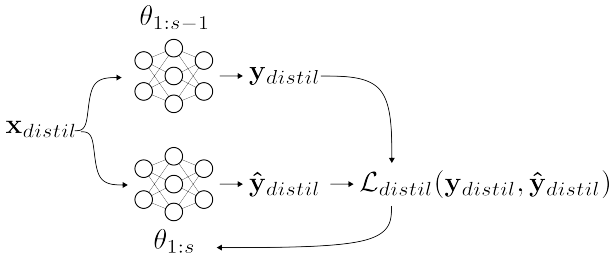


Fig. 6: Schematic representation of our version of knowledge distillation. $\theta_{1:s}$ denotes the currently training model. $\theta_{1:s-1}$ is used for the saved model.

MLP. This way, the signal the MLP needs to learn can be low frequency, with the mapping being arbitrarily high frequency. As the mapping is learnt instead of designed, it becomes very well suited to the specific signal to be learnt. However, because the encoding has to be learnt, the mappings for points which have not been trained on are unknown, making DINER unsuitable for interpolation.

B. Knowledge Distillation for Neural Fields

As mentioned in the related work, what input to use for knowledge distillation is complicated, as the input to the model is often complex and large to store. However, as the input for neural fields is more structured, we use this structure to generate the data used for distillation.

After training a task, the model parameters are stored. These parameters are replaced whenever a task finishes training, so only one set of previous parameters is stored. Additionally, if the input is expanded, after training a task, we store the domain of the coordinates of that task. This is done by storing, for each axis of the coordinate space, the highest and lowest coordinate value.

When training a subsequent task, points are sampled uniformly at random from the domains of the previous tasks. This input is called $\mathbf{x}_{distil}$. The number of points sampled is equal to a percentage p_{distil} of the number of points used to train the current task. If the input is expanded, this number of points is sampled for each task's domain. However, if the output is expanded, this number of points is sampled from the full range of the input for the model.

Then, an output is generated for $\mathbf{x}_{distil}$ using the previously trained and currently training parameters, called $\mathbf{y}_{distil}$ and $\hat{\mathbf{y}}_{distil}$, respectively. These outputs are used to calculate a loss, denoted as $\mathcal{L}_{distil}$, using the same loss function employed for fitting the task. This loss is combined with $\mathcal{L}_{fit}$ as the loss function minimised during training, in the following manner:

$$\mathcal{L}_{total} = \mathcal{L}_{fit} + \lambda \mathcal{L}_{distil}$$

Here, λ is a hyperparameter used to tune the importance of distillation and compensate for differences in magnitude for losses of different tasks.

A schematic representation of our method is shown in Figure 6.

C. DINER Stabilisation

A DINER model consists of two parts, each serving a distinct purpose. The hash map, which maps input coordinates to a latent space, represents the structure of each point that has been learned. The MLP backbone learns a mapping from the latent space to the output space.

This explicit separation between learning the structure of the signal and learning how to represent the signal can be used to prevent forgetting. When expanding the input space used, the values of the previously learnt points in the hashmap will not change during training. Therefore, if only the hashmap is trained and the MLP is frozen after learning the first task, no forgetting can occur. If the distribution of the signal is similar between tasks, the MLP should also be able to represent the later tasks.

Similarly, when adding another output to the model, the structure at each point remains the same, while the mapping changes only slightly. Therefore, by freezing the hashmap and only learning the MLP, the learnt function from the latent space to the original output space has the same input. If the original signal is a good predictor for the new signals, learning the new signal should also work well.

In summary, when the input is expanded, the MLP is frozen, while if the signal is expanded, the hashtable is frozen.

D. Datasets

In our experiments, three datasets are used. Two datasets focus on input expansion: the ImageNet and 4D ACDC datasets. The ACDC Segmentation dataset is used to test output expansion.

The representations for all datasets are learnt using Huber loss [37].

1) ImageNet: Images from the ImageNet [38] dataset are resampled such that the shortest axis has 256 pixels. Then, centre crops are taken, resulting in 256x256 pixel images. These images are vertically split into four 256x64 bars. These bars are used as the tasks. This is done for 25 images.

To fit these images, the RGB values of the images are scaled to be between 0 and 1. The MLP of the neural field has three outputs, corresponding to each of the colour channels. A sigmoid activation function is added to the output of the model.

2) 4D ACDC: In this dataset, 4D MRI scans from the ACDC dataset [39] are used for representation. Each of these scans consists of a sequence of 3D frames which together show one complete heart cycle. From each series of scans, twelve scans are selected. This is done such that the entire

heart cycle is represented and all scans are as evenly spaced as possible. Twelve scans are used, as this is the length of the shortest sequence. Each of these frames is learnt sequentially, making up the tasks. This has been done for 10 scans.

To fit the MR scans, the voxel intensity values are normalised to lie between 0 and 1. The MLP of the neural field has one output, which does not use an activation function.

3) ACDC segmentation: In this test, MRI scans and segmentations from the ACDC are learnt in two tasks in an output expansion way. First, the full MR scan is taught to the model. Then the full segmentation mask is given to the model. This segmentation mask consists of the left ventricular endocardium and epicardium, as well as the right ventricular endocardium. This has been done for 10 scan and segmentation combinations.

The voxel intensities are again normalised between 0 and 1. The neural field has five outputs, corresponding to the scan representation, the background, and the segmentation classes. Each output has a sigmoid activation function. To fit the segmentation mask, binary cross-entropy loss is used.

VI. Experiments and Results

We test our setups on 2D image representation and various forms of MRI representation.

In this section, several graphs are presented that compare performance on the first and last tasks. In these graphs, models in the top right corner have the least forgetting and can fit the tasks well. In the top left are models that perform well on the last task, but suffer heavily from forgetting. In the bottom right are models that maintain good performance but suffer from low plasticity. Finally, in the bottom left are models that perform poorly, either due to a combination of forgetting and low plasticity or because they are unable to fit the signal well. Additionally, attention should be paid to the axis labels, as there is sometimes a difference in scale between the x and y axes.

The performance on the first and last tasks, after the model finished training on the last task, is reported. Both are needed, as some training methods are more susceptible to forgetting, while others are more susceptible to low plasticity. These are also used to illustrate the differences in how well the models can fit the signals, which vary between models and training methods.

For the tables, the mean value is given, along with the standard deviation in brackets.

A. Experimental Settings

All models employed the same basic layout, consisting of three hidden layers with 256 neurons each. For image reconstruction tasks, the PSNR and SSIM [40] are reported. For segmentation tasks, the mean Dice similarity index is given for all non-background tasks.

B. Model Hyperparameter Experiment

In this test, the effect of model hyperparameters on the learning and forgetting performance is evaluated. This means that no continual learning strategies are employed. For this test, the ImageNet dataset will be used, as described above.

TABLE II: Comparison of first and last task performance on the ImageNet dataset by model. The chosen model hyperparameters have a big impact on the continual learning behaviour.

Model	Hyperparameters	Task 1		Task 4	
		PSNR	SSIM	PSNR	SSIM
P.E. ReLU	L=5	10.8 (4.03)	0.19 (0.11)	25.7 (2.56)	0.70 (0.10)
	L=10	11.1 (4.16)	0.17 (0.10)	26.6 (4.29)	0.73 (0.17)
SIREN	$\omega_0 = 15, \omega_i = 1$	13.5 (2.23)	0.31 (0.17)	23.5 (2.70)	0.54 (0.17)
	$\omega_0 = 15, \omega_i = 15$	11.7 (3.13)	0.08 (0.04)	35.1 (2.43)	0.95 (0.02)
	$\omega_0 = 30, \omega_i = 1$	12.2 (2.44)	0.22 (0.12)	25.5 (2.67)	0.67 (0.13)
	$\omega_0 = 30, \omega_i = 30$	11.1 (3.98)	0.07 (0.04)	38.7 (7.30)	0.93 (0.18)
FINER	$\omega_0 = 5, \omega_i = 1$	12.6 (2.48)	0.07 (0.03)	39.5 (1.85)	0.98 (0.02)
	$\omega_0 = 5, \omega_i = 5$	11.8 (3.47)	0.07 (0.05)	46.2 (2.44)	0.99 (0.01)
DINER	L=1	16.0 (5.88)	0.64 (0.25)	28.8 (8.48)	0.81 (0.32)
	L=2	16.1 (5.60)	0.66 (0.27)	36.8 (11.0)	0.88 (0.30)
	L=3	16.6 (5.74)	0.71 (0.23)	44.8 (13.3)	0.92 (0.25)

In Figure 7, the results of reconstructions for a sample image from all models are compared.

Figure 8 and Table II present a comparison of the first and last task performance, in terms of both PSNR and SSIM, for all models and their respective hyperparameters. The full PSNR and SSIM performance is in Table IX. From this and the comparison graph, it can be seen that the model choice has a significant influence on both fitting performance and continual learning behaviour.

The P.E. ReLU suffers heavily from forgetting. In this case, it appears to be due to the similarity of the input, which causes the model to repeat the last task for earlier tasks. This is especially visible in the reconstructed image, where the bottom of the fish is visible for the earlier tasks.

For SIREN, the choice of ω_0 and ω_i influences the performance. If ω_i is set to 1, the model appears to retain a vague memory of the previous tasks' structure. This is evident in the SSIM, but particularly in Figure 7, where parts of the fish remain distinguishable after the task switch. However, when ω_i is set to be equal to ω_0, the output from previous tasks more closely resembles noise than the structure that was there before. On the other hand, setting the ω_i to 1 instead of equal to ω_0 significantly lowers the fitting performance.

For FINER, setting ω_i to 1 instead of ω_0 has a lesser effect on both fitting and forgetting performance. Here, for both versions of the model, most of the previously learnt representation is forgotten after the training task switch.

For DINER, the colours output by the model are correct for old tasks, but the structure remains more stable after training on later tasks. This can be observed in the images, as well as in the difference between PSNR and SSIM for the models. All models maintain a relatively high SSIM, but the PSNR decreases more significantly. Additionally, as the latent space dimension increases, the colours are kept better as well.

An issue arises for DINER when incrementally learning an image, where the first task consists of an even colour. In one image of ImageNet, shown in Figure 9, the entire first task consists of an evenly coloured sky. When this is the case, the model is unable to process the rest of the image, instead consistently showing this even colour, even when being trained for other colours. This shows an extreme example of a lack of plasticity.

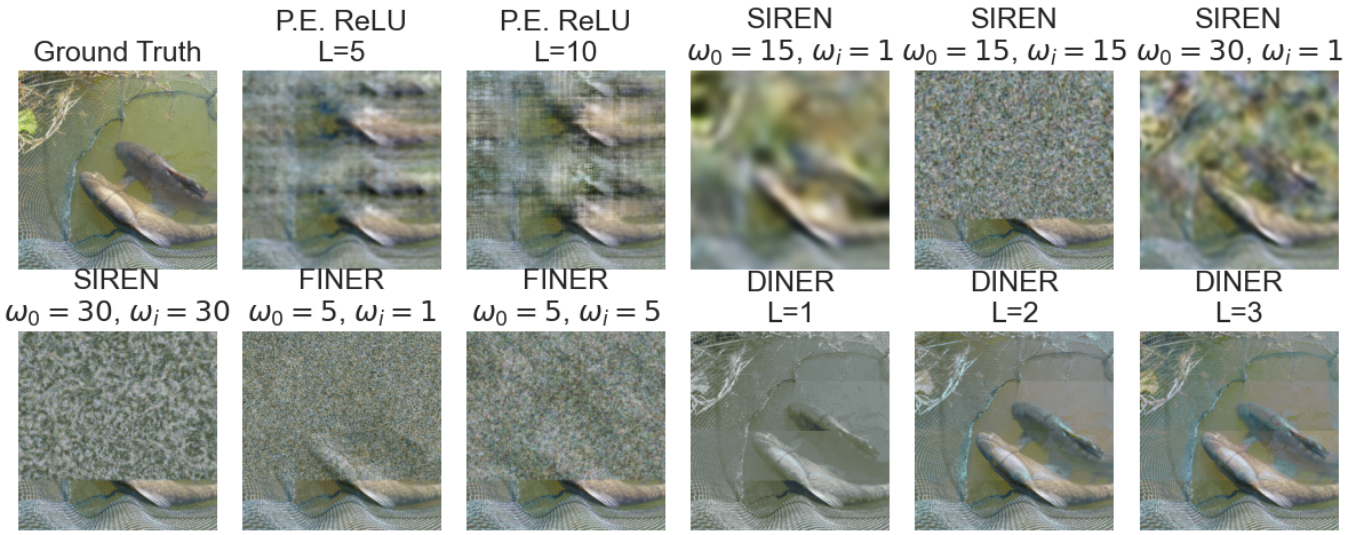


Fig. 7: Comparison of model outputs after training the last task on an example from the ImageNet dataset

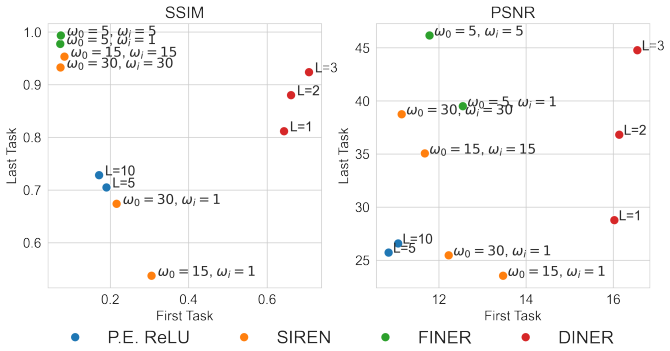


Fig. 8: Comparison of first and last task performance on the ImageNet dataset by model. The chosen model hyperparameters have a big impact on the continual learning behaviour.

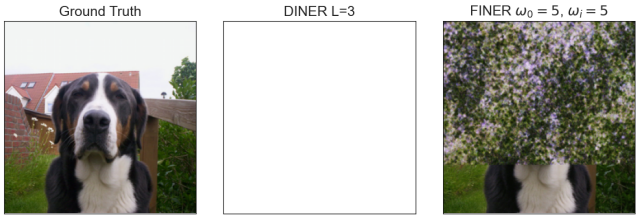
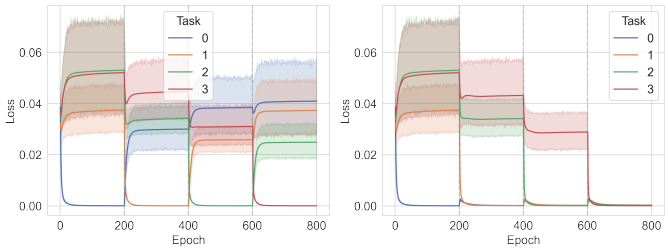


Fig. 9: Comparison of model outputs after training the last task on a sample in the ImageNet dataset. If the first task consists almost entirely of one colour, DINER does not learn to represent other colours. Other models like FINER do not suffer from this.

C. Distillation Experiments

In this experiment, samples are trained using distillation to demonstrate the influence on training performance. We use different values of p_{distil}, the ratio of distillation samples used. Distillation is compared against a benchmark of not applying any special technique, referred to as 'naive'.



(a) Naive (b) Distillation $p_{distil} = 20\%$

Fig. 10: Comparison of taskwise loss during training on the ImageNet dataset for FINER $\omega_0 = 5, \omega_i = 5$. It shows that training naively causes the loss of old tasks to rise, while distillation keeps it low.

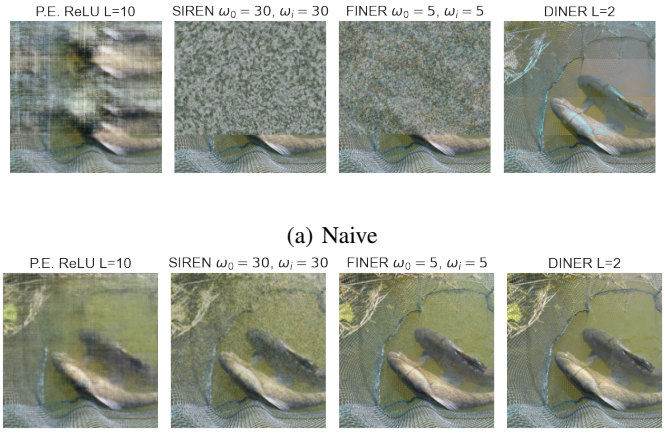
This test is conducted on the ImageNet, 4D ACDC, and ACDC Segmentation datasets.

1) ImageNet: In Figure 10, the mean per-task losses of a naively trained and a distillation-trained FINER model are compared for the ImageNet dataset. For the naive model, catastrophic forgetting is evident, as the loss of the previous task returns to an untrained level after training begins on a new task. However, for the distillation model, the loss increases only slightly before decreasing to a lower level.

In Figure 11, the input images are reconstructed after training for each model. These models are trained naively in Figure 11a and using distillation in Figure 11b. As is visible, all of the models visually more closely reconstruct the original image in both structure and colour when using distillation.

Similar results are also visible in Figure 12 and Table III, which compare the performance on the first and last tasks. The performance on all tasks is in Table X. For every model, using distillation increases both SSIM and PSNR on the first task compared to not using any strategy. The effect it has depends on the model.

For the positionally encoded model, both SSIM and PSNR



(a) Naive

(b) Distillation $p_{distil} = 20\%$

Fig. 11: Comparison of model outputs after training the last task on an example from the ImageNet dataset. Structure and colour are kept much better with distillation.

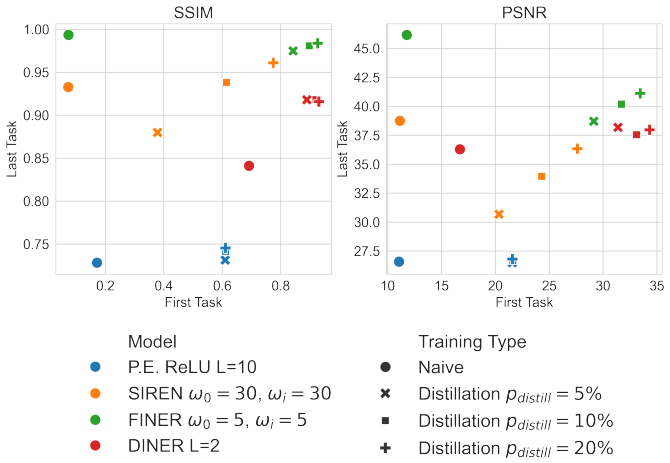


Fig. 12: Comparison of first and last task performance on the ImageNet dataset with distillation.

of the first task improve when distillation is applied. The number of samples used does not significantly impact the results.

SIREN and FINER have similar behaviour. Both lose performance on the last task when distillation is applied. Some of this performance is regained if more samples are used for distillation. The FINER models show these patterns slightly less.

For the DINER model, applying distillation improves performance not just on the earlier tasks, but also on the last task. This is notable, as it would be expected that the model performs best on the last task when no continual learning strategy is employed.

2) ACDC 4D: In Figure 13 and Table IV, the performance on the first and last scans in the sequences is compared. The performance on the first and last two out of twelve tasks is shown in Table XI. The effect of distillation on performance varies depending on the model.

TABLE III: Comparison of first and last task performance on the ImageNet dataset with distillation.

Model	Training Type	Task 1		Task 4	
		PSNR	SSIM	PSNR	SSIM
P.E. ReLU	Naive	11.1 (4.16)	0.17 (0.10)	26.6 (4.29)	0.73 (0.17)
	Distillation $p_{distill} = 5\%$	21.6 (2.86)	0.61 (0.12)	26.5 (2.75)	0.73 (0.09)
	Distillation $p_{distill} = 10\%$	21.6 (2.86)	0.61 (0.13)	26.7 (2.75)	0.74 (0.08)
	Distillation $p_{distill} = 20\%$	21.6 (2.81)	0.61 (0.12)	26.8 (2.74)	0.75 (0.08)
SIREN	Naive	11.1 (3.98)	0.07 (0.04)	38.7 (7.30)	0.93 (0.18)
	Distillation $p_{distill} = 5\%$	20.3 (4.06)	0.38 (0.18)	30.7 (3.88)	0.88 (0.07)
	Distillation $p_{distill} = 10\%$	24.3 (4.98)	0.61 (0.19)	33.9 (3.79)	0.94 (0.04)
	Distillation $p_{distill} = 20\%$	27.6 (5.28)	0.77 (0.13)	36.3 (3.36)	0.96 (0.02)
FINER	Naive	11.8 (3.47)	0.07 (0.05)	46.2 (2.44)	0.99 (0.01)
	Distillation $p_{distill} = 5\%$	29.1 (3.80)	0.84 (0.07)	38.7 (2.67)	0.97 (0.01)
	Distillation $p_{distill} = 10\%$	31.7 (3.72)	0.90 (0.05)	40.2 (2.49)	0.98 (0.01)
	Distillation $p_{distill} = 20\%$	33.5 (3.39)	0.93 (0.03)	41.1 (2.34)	0.98 (0.01)
DINER	Naive	16.7 (5.46)	0.69 (0.23)	36.3 (12.1)	0.84 (0.34)
	Distillation $p_{distill} = 5\%$	31.4 (5.02)	0.89 (0.19)	38.2 (9.94)	0.92 (0.25)
	Distillation $p_{distill} = 10\%$	33.1 (5.92)	0.92 (0.19)	37.6 (9.75)	0.92 (0.25)
	Distillation $p_{distill} = 20\%$	34.3 (6.42)	0.93 (0.19)	38.0 (10.1)	0.92 (0.25)

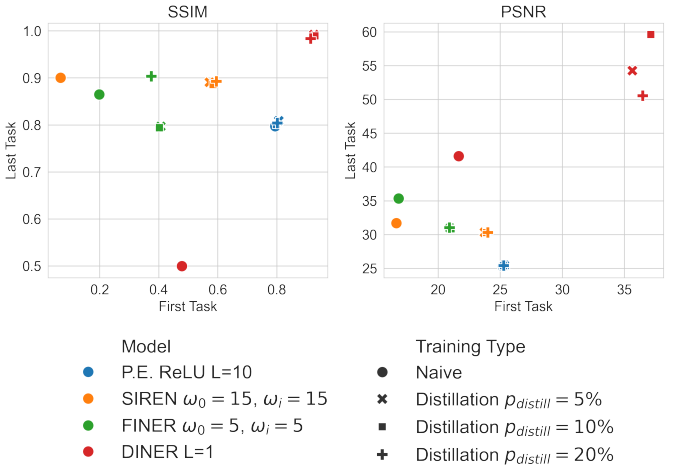


Fig. 13: Comparison of first and last task performance on the ACDC 4D dataset with distillation.

For the positionally encoded model, applying distillation does not significantly influence the continual learning behaviour. However, this is also the case because the positionally encoded model does not seem to forget or suffer from low plasticity to a significant extent for this dataset.

For SIREN and FINER, applying distillation improves performance on the old tasks, but as the task becomes older, performance is lost further. The number of samples used for distillation does not significantly change this.

Without distillation, DINER performs poorly on all tasks, including the last one. However, when some distillation is applied, the model becomes much more capable of fitting the data.

3) ACDC Segmentation: Since every model fits the segmentation mask well, Figure 14 instead shows the performance on the representation portion to illustrate forgetting. This is done through a comparison of the representation metrics before and after training the segmentation mask. In this graph, the x-axis shows the model's ability to fit the scan, while the y-axis indicates the model's capacity to keep its representation. The Dice scores, PSNR, and SSIM after training are shown in Table V.

For the positional encoded, SIREN, and FINER models, applying distillation significantly improves performance on the MRI representation. This is done without decreasing perfor-

TABLE IV: Comparison of first and last task performance on the ACDC 4D dataset with distillation.

Model	Training Type	Task 1		Task 12	
		PSNR	SSIM	PSNR	SSIM
P.E. ReLU	Naive	25.2 (2.96)	0.79 (0.05)	25.3 (2.94)	0.80 (0.05)
	Distillation $p_{distill} = 5\%$	25.3 (3.28)	0.81 (0.05)	25.4 (3.28)	0.81 (0.05)
	Distillation $p_{distill} = 10\%$	25.3 (3.41)	0.80 (0.06)	25.4 (3.43)	0.80 (0.06)
	Distillation $p_{distill} = 20\%$	25.3 (3.44)	0.80 (0.05)	25.4 (3.44)	0.80 (0.05)
SIREN	Naive	16.6 (2.99)	0.07 (0.04)	31.7 (4.26)	0.90 (0.07)
	Distillation $p_{distill} = 5\%$	23.7 (3.51)	0.57 (0.08)	30.3 (3.68)	0.89 (0.06)
	Distillation $p_{distill} = 10\%$	23.9 (3.51)	0.58 (0.08)	30.3 (3.96)	0.89 (0.07)
	Distillation $p_{distill} = 20\%$	24.0 (3.34)	0.60 (0.07)	30.3 (3.59)	0.89 (0.06)
FINER	Naive	16.8 (3.46)	0.20 (0.06)	35.3 (3.58)	0.86 (0.05)
	Distillation $p_{distill} = 5\%$	21.0 (2.91)	0.41 (0.07)	30.9 (2.24)	0.80 (0.04)
	Distillation $p_{distill} = 10\%$	21.0 (2.90)	0.40 (0.07)	30.9 (1.99)	0.79 (0.04)
	Distillation $p_{distill} = 20\%$	20.9 (3.02)	0.38 (0.05)	31.0 (1.78)	0.90 (0.02)
DINER	Naive	21.7 (5.17)	0.48 (0.40)	41.6 (41.5)	0.50 (0.43)
	Distillation $p_{distill} = 5\%$	35.6 (5.48)	0.93 (0.05)	54.2 (9.60)	0.99 (0.01)
	Distillation $p_{distill} = 10\%$	37.1 (6.73)	0.93 (0.05)	59.6 (20.9)	0.99 (0.01)
	Distillation $p_{distill} = 20\%$	36.5 (6.00)	0.92 (0.07)	50.5 (9.88)	0.98 (0.01)

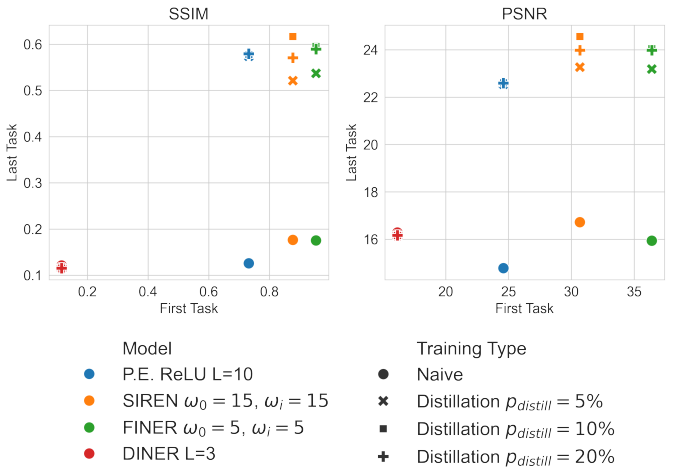


Fig. 14: Comparison of representation performance after training just the first and both second tasks on the ACDC Segmentation dataset with distillation.

mance on the segmentation representation task. The extent to which the model can keep its representation of the scan varies depending on the model, with the SIREN model performing best.

The DINER model struggles with representing the scan when trained in this manner, even when trained solely on it. Because of this, no forgetting takes place, so the distillation has no effect on the forgetting behaviour. However, the model does accurately represent the segmentation mask.

D. DINER stabilisation

In this experiment, models are trained using stabilisation of a part of a DINER model to show the influence on training performance. The MLP is frozen for input expansion, while the hash map is frozen for output expansion. The stabilisation strategy is compared to distillation with $p_{distil} = 20\%$ and naively training. This approach is specific to the DINER architecture, as DINER decouples the encoding and representation parts of the model. This test is done on the ImageNet, 4D ACDC, and ACDC segmentation datasets.

1) ImageNet: In Figure 15, the losses of naive training, training with distillation, and stabilised training are compared. There, it is visible that the stabilised model does not lose performance on old tasks when training new tasks, thereby

TABLE V: Comparison of performance on the ACDC Segmentation dataset with distillation.

Model	Training Type	Mean Dice	PSNR	SSIM
P.E. ReLU	Naive	0.95 (0.03)	14.8 (1.57)	0.13 (0.06)
	Distillation $p_{distill} = 5\%$	0.95 (0.03)	22.5 (1.64)	0.57 (0.02)
	Distillation $p_{distill} = 10\%$	0.95 (0.03)	22.6 (1.64)	0.58 (0.02)
	Distillation $p_{distill} = 20\%$	0.95 (0.03)	22.6 (1.62)	0.58 (0.02)
SIREN	Naive	1.00 (0.01)	16.7 (1.90)	0.18 (0.05)
	Distillation $p_{distill} = 5\%$	0.95 (0.10)	23.3 (2.45)	0.52 (0.14)
	Distillation $p_{distill} = 10\%$	1.00 (0.00)	24.6 (1.16)	0.62 (0.06)
	Distillation $p_{distill} = 20\%$	0.97 (0.07)	24.0 (1.84)	0.57 (0.16)
FINER	Naive	1.00 (0.00)	15.9 (1.78)	0.18 (0.06)
	Distillation $p_{distill} = 5\%$	1.00 (0.00)	23.2 (1.23)	0.54 (0.07)
	Distillation $p_{distill} = 10\%$	1.00 (0.00)	24.0 (1.05)	0.60 (0.06)
	Distillation $p_{distill} = 20\%$	1.00 (0.00)	24.0 (1.35)	0.59 (0.06)
DINER	Naive	1.00 (0.00)	16.3 (1.99)	0.12 (0.07)
	Distillation $p_{distill} = 5\%$	1.00 (0.01)	16.2 (1.96)	0.12 (0.07)
	Distillation $p_{distill} = 10\%$	0.99 (0.04)	16.2 (1.96)	0.12 (0.07)
	Distillation $p_{distill} = 20\%$	1.00 (0.00)	16.2 (1.96)	0.12 (0.07)

TABLE VI: Comparison of performance on the Imagenet dataset with stabilisation.

Training Type	Task 1		Task 4	
	PSNR	SSIM	PSNR	SSIM
Naive	16.7 (5.46)	0.69 (0.23)	36.3 (12.1)	0.84 (0.34)
Distillation $p_{distill} = 20\%$	34.3 (6.42)	0.93 (0.19)	38.0 (10.1)	0.92 (0.25)
Stabilised	36.6 (7.51)	0.94 (0.20)	30.6 (9.18)	0.88 (0.24)

preventing forgetting. It is also clear that performance for later tasks does not reach the level of performance that earlier tasks had immediately after training. However, this is also the case for the naive trained models and the models trained with distillation. This shows that the DINER model itself suffers from a lack of plasticity.

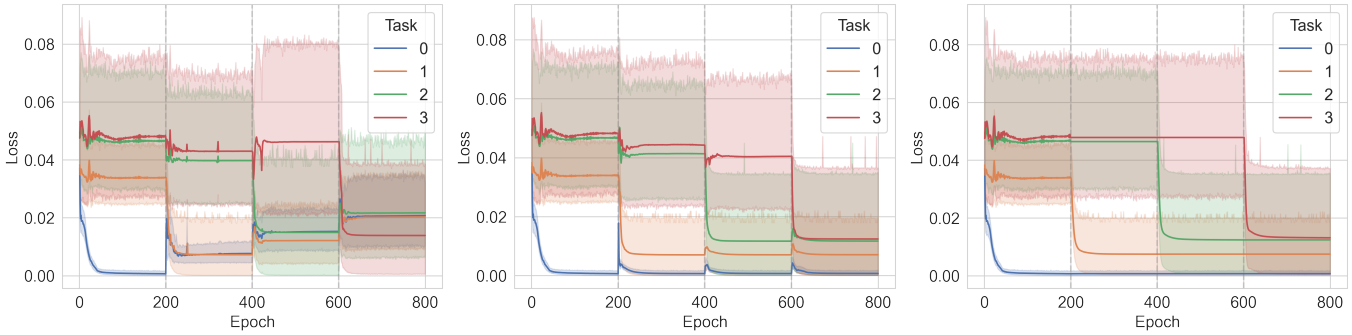
Figure 16 compares an example of the outputs of the naive trained, distillation trained and stabilised trained models. The naive sample suffers from a colour representation that changes when the tasks change, which means the colours for older tasks are not correct. The distillation and stabilised models do not suffer from this.

Also in the SSIM and especially the PSNR values of Figure 17 and Table VI, this pattern is visible. While the SSIM remains relatively high when training naively, the PSNR drops significantly, indicating that the colours are not accurate, while the structure is. For models trained with distillation or stabilisation, both the PSNR and SSIM are higher overall. However, for the distillation models, the later tasks have a higher SSIM and PSNR, while for the stabilised models, this is the case for the earlier tasks.

Both distillation and stabilisation perform well on the ImageNet dataset, and which performs best depends on whether stability or plasticity is more important. For stability, stabilisation performs best, while for plasticity, distillation performs best.

2) ACDC 4D: For the ACDC 4D dataset, the difference is bigger. In Figure 18 and Table VII, the SSIM and PSNR of the first and last two tasks are compared. Here, it is evident that the stabilised model can represent later tasks well using the representation learnt in the first task. It performs better than the model trained using distillation in terms of PSNR, but the SSIM is slightly lower for later tasks.

3) ACDC Segmentation: The performance on the ACDC Segmentation dataset is shown in Figure 19 and Table VIII. Because the DINER model is unable to fit the representation well, it is difficult to determine the effect of stabilisation and distillation compared to naive training.



(a) Naive (b) Distillation $p_{distil} = 20\%$ (c) Stabilisation

Fig. 15: Comparison of taskwise loss during training on the ImageNet dataset for DINER L=2

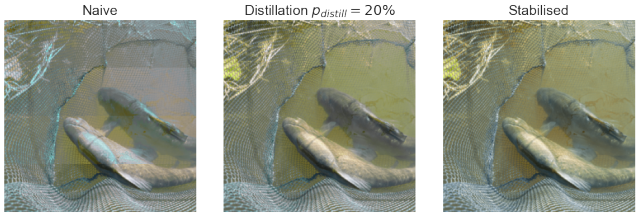


Fig. 16: Comparison of DINER L=2 outputs after training the last task on an example from the ImageNet dataset

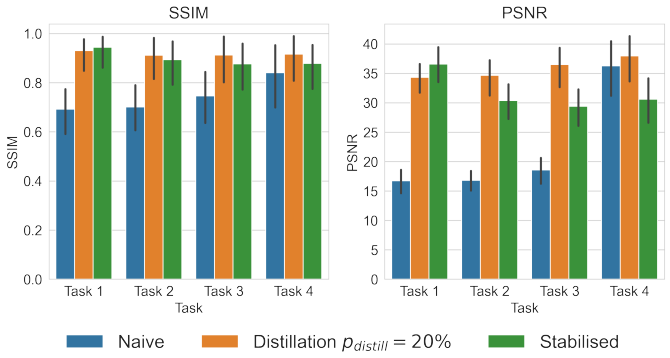


Fig. 17: Comparison of performance on the ImageNet dataset with stabilisation.

VII. Discussion

In this work, we demonstrated that neural fields are highly susceptible to catastrophic forgetting in a continual learning setting. We also demonstrated that the extent to which this issue arises depends on the model architecture used. Positionally encoded ReLU, SIREN, and FINER models are more susceptible to forgetting than DINER models.

To address this issue, we explored utilising the trained model to mitigate forgetting through knowledge distillation. We have demonstrated that knowledge distillation can mitigate a significant amount of forgetting in neural fields in both input and output expansion scenarios. Additionally, we have demonstrated that when the MLP backbone of a DINER model is frozen after training on the first task, the model can often learn to represent subsequent tasks when expanding the input regions trained on.

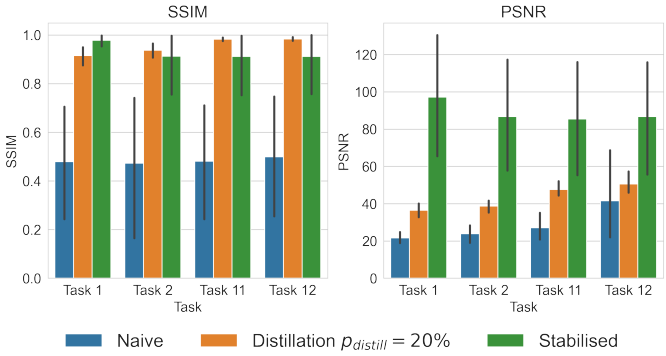


Fig. 18: Comparison of performance on the first and last two tasks on the ACDC 4D dataset with stabilisation.

TABLE VII: Comparison of first and last task performance on the ACDC 4D dataset with distillation.

Training Type	Task 1		Task 12	
	PSNR	SSIM	PSNR	SSIM
Naive	21.7 (5.17)	0.48 (0.40)	41.6 (41.5)	0.50 (0.43)
Distillation $p_{distill} = 20\%$	36.5 (6.00)	0.92 (0.07)	50.5 (9.88)	0.98 (0.01)
Stabilised	97.2 (55.1)	0.98 (0.04)	86.7 (52.5)	0.91 (0.24)

A. Model Hyperparamters Results

In this work, we have shown that for neural fields, the choice of architecture not only affects the model behaviour while fitting but also in a continual learning setting. This yields similar results to previous work, which has demonstrated that the choice of activation function affects the continual learning behaviour of a neural network [17].

Another notable result from the naive trained models is the effect of the ω_i values for the SIREN model. When this is set to 1, the model suffers a lot less from forgetting compared to when it is set to be equal to ω_0. This could be because having a high ω_i value means that a slight change in input to the activation function has a bigger difference for the output. As a result, changes in the model's weights have a greater impact on its output. This could also explain the difference between the SIREN models with $\omega_0 = 15$ and $\omega_0 = 30$.

Interestingly, the effect when changing ω_i is less visible for the FINER model. Here, both models perform similarly on old tasks, while the model with $\omega_i = 5$ outperforms the model with $\omega_i = 1$. This could be because the activation function

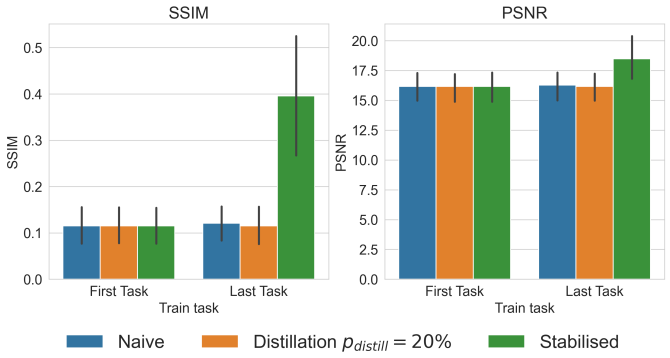


Fig. 19: Comparison of representation performance after training just the first and both second tasks on the ACDC Segmentation dataset with stabilisation.

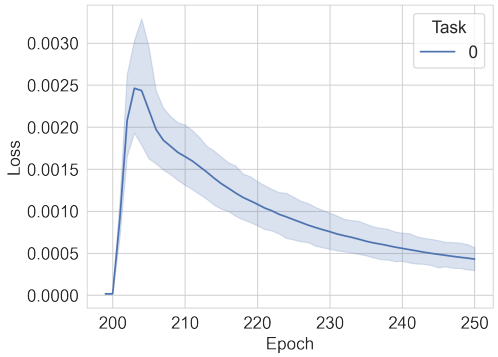
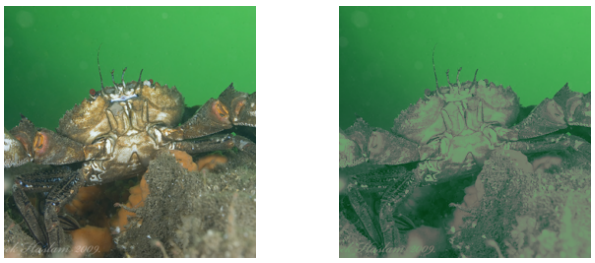


Fig. 20: First task loss during the switch to training the second task. The stability gap refers to the increase in loss on the first task that occurs when training for the second task commences.

TABLE VIII: Comparison of performance on the ACDC Segmentation dataset with stabilisation.

Training Type	Mean Dice	PSNR	SSIM
Naive	1.00 (0.00)	16.3 (1.99)	0.12 (0.07)
Distillation $p_{distill} = 20\%$	1.00 (0.00)	16.2 (1.96)	0.12 (0.07)
Stabilised	0.00 (0.00)	18.5 (2.94)	0.40 (0.22)



(a) Ground Truth (b) Stabilised Reconstruction

Fig. 21: Reconstruction of a sample under stabilisation for DINER L=2. Training stabilised can fail if the first task is dissimilar to the rest of the image.

used by FINER is already of a higher frequency. Therefore, a slight change in the model's weights might already result in such a significant difference.

Another interesting phenomenon is how DINER can keep the representation of the structure of the early tasks. This means that if the colours are similar between tasks, the colour representation learnt early can be used for later tasks.

B. Distillation Results

For the SIREN and FINER models, performance decreases on the last task when training with distillation. This is not an entirely unexpected phenomenon, following the stability-plasticity dilemma [41]. This means that some methods can prevent the model from forgetting, but do so by inhibiting learning. However, if this were the only factor, later task performance would continue to decrease as the model becomes more stable. This is not what happens, as when p_{distil} is increased and the model's performance becomes more stable, its later performance also increases.

A similar pattern is also visible for the DINER model, but to a lesser extent. The surprising part is that here, the performance on the last task is higher with distillation than with naive training. This is especially the case for the ACDC 4D dataset. This appears to be because the DINER model has low plasticity on its own.

In Figure 10b, the loss of previously trained tasks increases when a task shift appears, before decreasing to near the trained level. This initial loss spike is expanded in Figure 20. This indicates that some forgetting still occurs, but that distillation helps mitigate its effects. This behaviour has been identified before, being known as the 'stability gap' [42]. This refers to a loss of performance upon the start of a new task, which recovers as the new task is trained. Multiple factors could contribute to this behaviour. First, distillation works by comparing the outputs of the previously trained model with those of the current model. However, as these models are the same in the first epoch, there will be no difference between these outputs. So, the regularisation factor of distillation does not yet play a part. Next is the difference in scale between $\mathcal{L}_{fit}$ and $\mathcal{L}_{distil}$. As the loss for the current task starts high, while the distillation loss is low, lowering the overall loss is done best by decreasing the fitting loss. This can come at a small cost to the distillation loss. However, as the model starts to fit the current task better, lowering the overall loss is achieved most effectively by reducing both losses simultaneously. This then lowers the loss for the earlier tasks to near the previous level.

C. Stabilisation Results

The PSNR of later tasks for some samples from the ImageNet dataset is significantly lower for the stabilised model compared to the distillation model. An example of this can be seen in Figure 21. As is visible here, the model can reconstruct the first task, but for later tasks, while the structure is correct, the colours are wrong. This is because the first task does not allow the model to create a good representation of the output structure used across the tasks. Because of this, the model must attempt to approximate the later tasks without the colours present in these tasks. So, it is still able to learn the structure, but not the colours, which results in a higher SSIM, but a lower PSNR.

D. Future Work

In this study, we conducted an experimental evaluation of strategies that can be employed to enable neural fields to learn to represent signals incrementally. This phenomenon can also be studied theoretically. In recent years, research has been done to analyse catastrophic forgetting using the Neural Tangent Kernel (NTK). The central part of this method is the NTK itself. The NTK is a method for illustrating how the training dynamics of a model are influenced by the derivatives of the model output with respect to its parameters. This can be used to describe the training process for a neural network. From these analyses on catastrophic forgetting, two main routes can be identified.

The first angle is related to data relatedness. Doan et al. [12] uses the NTK to derive a formula for the forgetting occurring in neural networks during continual learning. An essential part of this formula is what they refer to as the 'NTK overlap matrix'. This overlap matrix represents the similarity of the NTK of a model across the inputs of different tasks. When these are similar, training on one task will have a significant influence on the output of the other tasks. So, to prevent forgetting, the overlap of the NTKs should be as small as possible.

When training neural fields, in contrast to other neural network approaches, the input to the model is more structured. To an extent, it can often be known how a neural field could be extended in the future, even if the measurements with which it has to be trained are not yet available. The knowledge of how the input is structured and control over how the input can be extended could be used to influence the overlap of the NTKs, thereby preventing forgetting. However, initial experiments with reducing the similarity of the input to the models have not yet yielded a significant improvement.

Lan and Mahmood [43] also identify the overlap between NTKs as the leading cause of forgetting, but propose a different solution to it. In their work, the authors hypothesise that this overlap can be minimised by increasing the sparsity of the model gradients. This is because when the gradients become sparser, the likelihood of overlap between the NTKs becomes smaller. To this end, the authors propose the elephant activation function, which is sparse and has a sparse gradient. Using this activation function in the penultimate layer of a neural network yields a performance improvement compared to naive training and EWC. However, initial tests in replacing the finer function of the second-to-last layer of a FINER network with the elephant function did not significantly increase the forgetting performance.

VIII. Conclusion

In this work, we have demonstrated that neural fields are severely affected by catastrophic forgetting in a continual learning setting, with SIREN and FINER models experiencing more significant issues than DINER models. Additionally, we have demonstrated that knowledge distillation and partial model freezing can, to some extent, mitigate these issues.

References

[1] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar, "Neural fields in visual computing and beyond," *Computer Graphics Forum*, vol. 41, no. 2, pp. 641–676, 2022, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14505. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14505

[2] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: learning continuous signed distance functions for shape representation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 165–174, iSSN: 2575-7075. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8954065

[3] Y. Sun, J. Liu, M. Xie, B. Wohlberg, and U. S. Kamilov, "CoIL: coordinate-based internal learning for tomographic imaging," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 1400–1412, 2021, conference Name: IEEE Transactions on Computational Imaging. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9606601

[4] A. W. Reed, H. Kim, R. Anirudh, K. A. Mohan, K. Champley, J. Kang, and S. Jayasuriya, "Dynamic CT reconstruction from limited views with implicit neural representations and parametric motion fields," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 2238–2248, iSSN: 2380-7504. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9709943

[5] L. Shen, J. Pauly, and L. Xing, "NeRP: implicit neural representation learning with prior embedding for sparsely sampled image reconstruction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 770–782, Jan. 2024, conference Name: IEEE Transactions on Neural Networks and Learning Systems. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9788018

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, pp. 99–106, Dec. 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3503250

[7] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the wild: neural radiance fields for unconstrained photo collections," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 7206–7215, iSSN: 2575-7075. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9578784

[8] Z. Chen, Y. Chen, J. Liu, X. Xu, V. Goel, Z. Wang, H. Shi, and X. Wang, "VideoINR: learning video implicit neural representation for continuous space-time super-resolution," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 2037–2047, iSSN: 2575-7075. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9878371

[9] D. Alblas, M. Hofman, C. Brune, K. K. Yeung, and J. M. Wolterink, "Implicit neural representations for modeling of abdominal aortic aneurysm progression," in *Functional Imaging and Modeling of the Heart*, O. Bernard, P. Clarysse, N. Duchateau, J. Ohayon, and M. Viallon, Eds. Cham: Springer Nature Switzerland, 2023, pp. 356–365.

[10] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: the sequential learning problem," in *Psychology of Learning and Motivation*, G. H. Bower, Ed. Academic Press, Jan. 1989, vol. 24, pp. 109–165. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0079742108605368

[11] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: theory, method and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 8, pp. 5362–5383, Aug. 2024, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. [Online]. Available: https://ieeexplore.ieee.org/ielx7/34/10582780/10444954/supp1-3367329.pdf?arnumber=10444954

[12] T. Doan, M. A. Bennani, B. Mazoure, G. Rabusseau, and P. Alquier, "A theoretical analysis of catastrophic forgetting through the NTK overlap matrix," in *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2021, pp. 1072–1080, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v130/doan21a.html

[13] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, May 2019, pp. 5301–5310, iSSN: 2640-3498 shortConferenceName: ICML. [Online]. Available: https://proceedings.mlr.press/v97/rahaman19a.html

[14] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit Neural Representations with Periodic Activation Functions," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 7462–7473. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/53c04118df112c13a8c34b38343b9c10-Abstract.html

[15] Z. Liu, H. Zhu, Q. Zhang, J. Fu, W. Deng, Z. Ma, Y. Guo, and X. Cao, "FINER: flexible spectral-bias tuning in implicit NEural representation by variableperiodic activation functions," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 2713–2722, iSSN: 2575-7075. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10655302

[16] V. Saragadam, D. LeJeune, J. Tan, G. Balakrishnan, A. Veeraraghavan, and R. G. Baraniuk, "WIRE: wavelet implicit neural representations," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 18 507–18 516, iSSN: 2575-7075. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10204916

[17] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," Mar. 2015, arXiv:1312.6211 [stat]. [Online]. Available: http://arxiv.org/abs/1312.6211

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017, publisher: Proceedings of the National Academy of Sciences. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1611835114

[19] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, number: 1. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11651

[20] S. Dohare, J. F. Hernandez-Garcia, Q. Lan, P. Rahman, A. R. Mahmood, and R. S. Sutton, "Loss of plasticity in deep continual learning," *Nature*, vol. 632, no. 8026, pp. 768–774, Aug. 2024, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41586-024-07711-7

[21] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, Jul. 2017, pp. 3987–3995, iSSN: 2640-3498 shortConferenceName: ICML. [Online]. Available: https://proceedings.mlr.press/v70/zenke17a.html

[22] L. Kumari, S. Wang, T. Zhou, and J. A. Bilmes, "Retrospective adversarial replay for continual learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 28 530–28 544, Dec. 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/b6ffbbacbe2e56f2ec9a0da907382b4a-Abstract-Conference.html

[23] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia, "Online continual learning with maximal interfered retrieval," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/15825aee15eb335cc13f9b559f166ee8-Abstract.html

[24] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: incremental classifier and representation learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5533–5542, iSSN: 1063-6919. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8100070

[25] K. Lee, K. Lee, J. Shin, and H. Lee, "Overcoming catastrophic forgetting with unlabeled data in the wild," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019, pp. 312–321, iSSN: 2380-7504. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9010368

[26] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8107520

[27] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. PMLR, Jun. 2020, pp. 3762–3773, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v108/farajtabar20a.html

[28] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 4548–4557, iSSN: 2640-3498 shortConferenceName: ICML. [Online]. Available: https://proceedings.mlr.press/v80/serra18a.html

[29] S. Ebrahimi, F. Meier, R. Calandra, T. Darrell, and M. Rohrbach, "Adversarial continual learning," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 386–402.

[30] Z. Yan, Y. Tian, X. Shi, P. Guo, P. Wang, and H. Zha, "Continual neural mapping: learning an implicit scene representation from sequential observations," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 15 762–15 772, iSSN: 2380-7504. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9711071

[31] R. Po, Z. Dong, A. W. Bergman, and G. Wetzstein, "Instant continual learning of neural radiance fields," in *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, Oct. 2023, pp. 3326–3336, iSSN: 2473-9944. [Online]. Available: https://ieeexplore.ieee.org/document/10350630

[32] Z. Cai and M. Müller, "CLNeRF: continual learning meets NeRF," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023, pp. 23 128–23 137, iSSN: 2380-7504. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10377110

[33] J. Chung, K. Lee, S. Baik, and K. M. Lee, "MEIL-NeRF: memory-efficient incremental learning of neural radiance fields," *IEEE Access*, pp. 1–1, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/11029250

[34] M. Guo, C. Li, H. Chen, and G. H. Lee, "UNIKD: UNcertainty-filtered incremental knowledge distillation for neural implicit representation," in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds. Cham: Springer Nature Switzerland, 2025, pp. 237–254.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[36] H. Zhu, S. Xie, Z. Liu, F. Liu, Q. Zhang, Y. Zhou, Y. Lin, Z. Ma, and X. Cao, "Disorder-invariant implicit neural representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 8, pp. 5463–5478, Aug. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10436706

[37] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, Mar. 1964, publisher: Institute of Mathematical Statistics. [Online]. Available: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-35/issue-1/Robust-Estimation-of-a-Location-Parameter/10.1214/aoms/1177703732.full

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: a large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255, iSSN: 1063-6919. [Online]. Available: https://ieeexplore.ieee.org/document/5206848

[39] O. Bernard, A. Lalande, C. Zotti, F. Cervenansky, X. Yang, P.-A. Heng, I. Cetin, K. Lekadir, O. Camara, M. A. Gonzalez Ballester, G. Sanroma, S. Napel, S. Petersen, G. Tziritas, E. Grinias, M. Khened, V. A. Kollerathu, G. Krishnamurthi, M.-M. Rohé, X. Pennec, M. Sermesant, F. Isensee, P. Jäger, K. H. Maier-Hein, P. M. Full, I. Wolf, S. Engelhardt, C. F. Baumgartner, L. M. Koch, J. M. Wolterink, I. Išgum, Y. Jang, Y. Hong, J. Patravali, S. Jain, O. Humbert, and P.-M. Jodoin, "Deep learning techniques for automatic MRI cardiac multi-structures segmentation and diagnosis: is the problem solved?" *IEEE Transactions on Medical Imaging*, vol. 37, no. 11, pp. 2514–2525, Nov. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8360453

[40] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004. [Online]. Available: https://ieeexplore.ieee.org/document/1284395

[41] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers in Psychology*, vol. 4, Aug. 2013, publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2013.00504/full

[42] M. D. Lange, G. v. d. Ven, and T. Tuytelaars, "Continual evaluation for lifelong learning: identifying the stability gap," Mar. 2023, arXiv:2205.13452. [Online]. Available: http://arxiv.org/abs/2205.13452

[43] Q. Lan and A. R. Mahmood, “Elephant neural networks: born to Be a continual learner,” Oct. 2023, arXiv:2310.01365 [cs]. [Online]. Available: http://arxiv.org/abs/2310.01365

TABLE IX: Model Hyperparameter Experiment Results.

Model	Hyperparameters	Task 1		Task 2		Task 3		Task 4	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
P.E. ReLU	L=5	10.8 (4.03)	0.19 (0.11)	11.2 (2.80)	0.19 (0.09)	13.3 (2.92)	0.28 (0.09)	25.7 (2.56)	0.70 (0.10)
	L=10	11.1 (4.16)	0.17 (0.10)	11.0 (2.95)	0.18 (0.08)	13.4 (3.62)	0.27 (0.11)	26.6 (4.29)	0.73 (0.17)
SIREN	$\omega_0 = 15, \omega_i = 1$	13.5 (2.23)	0.31 (0.17)	14.9 (2.25)	0.36 (0.11)	17.4 (2.69)	0.44 (0.11)	23.5 (2.70)	0.54 (0.17)
	$\omega_0 = 15, \omega_i = 15$	11.7 (3.13)	0.08 (0.04)	11.8 (2.11)	0.07 (0.04)	12.4 (2.14)	0.07 (0.03)	35.1 (2.43)	0.95 (0.02)
	$\omega_0 = 30, \omega_i = 1$	12.2 (2.44)	0.22 (0.12)	14.2 (2.34)	0.32 (0.09)	15.9 (2.44)	0.40 (0.09)	25.5 (2.67)	0.67 (0.13)
	$\omega_0 = 30, \omega_i = 30$	11.1 (3.98)	0.07 (0.04)	11.0 (2.75)	0.07 (0.03)	11.9 (2.87)	0.07 (0.04)	38.7 (7.30)	0.93 (0.18)
FINER	$\omega_0 = 5, \omega_i = 1$	12.6 (2.48)	0.07 (0.03)	13.4 (1.71)	0.13 (0.04)	15.4 (2.26)	0.22 (0.07)	39.5 (1.85)	0.98 (0.02)
	$\omega_0 = 5, \omega_i = 5$	11.8 (3.47)	0.07 (0.05)	11.9 (2.54)	0.10 (0.05)	13.8 (3.08)	0.16 (0.06)	46.2 (2.44)	0.99 (0.01)
DINER	L=1	16.0 (5.88)	0.64 (0.25)	15.5 (4.90)	0.59 (0.29)	17.6 (6.08)	0.65 (0.30)	28.8 (8.48)	0.81 (0.32)
	L=2	16.1 (5.60)	0.66 (0.27)	16.4 (5.03)	0.66 (0.27)	17.9 (6.04)	0.68 (0.33)	36.8 (11.0)	0.88 (0.30)
	L=3	16.6 (5.74)	0.71 (0.23)	17.2 (5.15)	0.72 (0.24)	19.9 (5.80)	0.77 (0.27)	44.8 (13.3)	0.92 (0.25)

TABLE X: Imagenet Distillation Results

Model	Training Type	Task 1		Task 2		Task 3		Task 4	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
P.E. ReLU L=10	Naive	11.1 (4.16)	0.17 (0.10)	11.0 (2.95)	0.18 (0.08)	13.4 (3.62)	0.27 (0.11)	26.6 (4.29)	0.73 (0.17)
	Distillation $p_{distill} = 5\%$	21.6 (2.86)	0.61 (0.12)	22.2 (2.57)	0.60 (0.09)	23.7 (2.36)	0.65 (0.08)	26.5 (2.75)	0.73 (0.09)
	Distillation $p_{distill} = 10\%$	21.6 (2.86)	0.61 (0.13)	22.2 (2.57)	0.60 (0.09)	23.9 (2.32)	0.65 (0.08)	26.7 (2.75)	0.74 (0.08)
	Distillation $p_{distill} = 20\%$	21.6 (2.81)	0.61 (0.12)	22.3 (2.56)	0.60 (0.08)	23.9 (2.28)	0.65 (0.08)	26.8 (2.74)	0.75 (0.08)
SIREN $\omega_0 = 30, \omega_i = 30$	Naive	11.1 (3.98)	0.07 (0.04)	11.0 (2.75)	0.07 (0.03)	11.9 (2.87)	0.07 (0.04)	38.7 (7.30)	0.93 (0.18)
	Distillation $p_{distill} = 5\%$	20.3 (4.06)	0.38 (0.18)	21.4 (2.74)	0.53 (0.14)	25.1 (2.97)	0.71 (0.12)	30.7 (3.88)	0.88 (0.07)
	Distillation $p_{distill} = 10\%$	24.3 (4.98)	0.61 (0.19)	25.0 (3.69)	0.71 (0.14)	28.2 (3.82)	0.81 (0.10)	33.9 (3.79)	0.94 (0.04)
	Distillation $p_{distill} = 20\%$	27.6 (5.28)	0.77 (0.13)	27.9 (4.00)	0.81 (0.11)	30.7 (3.75)	0.88 (0.08)	36.3 (3.36)	0.96 (0.02)
FINER $\omega_0 = 5, \omega_i = 5$	Naive	11.8 (3.47)	0.07 (0.05)	11.9 (2.54)	0.10 (0.05)	13.8 (3.08)	0.16 (0.06)	46.2 (2.44)	0.99 (0.01)
	Distillation $p_{distill} = 5\%$	29.1 (3.80)	0.84 (0.07)	30.2 (2.89)	0.88 (0.05)	32.9 (2.73)	0.92 (0.04)	38.7 (2.67)	0.97 (0.01)
	Distillation $p_{distill} = 10\%$	31.7 (3.72)	0.90 (0.05)	32.3 (2.97)	0.92 (0.04)	34.4 (2.63)	0.94 (0.03)	40.2 (2.49)	0.98 (0.01)
	Distillation $p_{distill} = 20\%$	33.5 (3.39)	0.93 (0.03)	33.7 (2.76)	0.94 (0.03)	35.4 (2.39)	0.95 (0.03)	41.1 (2.34)	0.98 (0.01)
DINER L=2	Naive	16.7 (5.46)	0.69 (0.23)	16.8 (4.56)	0.70 (0.23)	18.6 (5.79)	0.75 (0.28)	36.3 (12.1)	0.84 (0.34)
	Distillation $p_{distill} = 5\%$	31.4 (5.02)	0.89 (0.19)	32.2 (7.53)	0.89 (0.24)	33.1 (8.10)	0.89 (0.25)	38.2 (9.94)	0.92 (0.25)
	Distillation $p_{distill} = 10\%$	33.1 (5.92)	0.92 (0.19)	33.7 (8.01)	0.90 (0.24)	35.0 (8.68)	0.91 (0.25)	37.6 (9.75)	0.92 (0.25)
	Distillation $p_{distill} = 20\%$	34.3 (6.42)	0.93 (0.19)	34.6 (8.29)	0.91 (0.24)	36.5 (8.95)	0.91 (0.25)	38.0 (10.1)	0.92 (0.25)

TABLE XI: ACDC 4D Distillation Results. For readability, only the first and last two tasks are included.

Model	Training Type	Task 1		Task 2		Task 11		Task 12	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
P.E. ReLU L=10	Naive	25.2 (2.96)	0.79 (0.05)	19.3 (3.99)	0.46 (0.08)	22.2 (2.69)	0.63 (0.12)	25.3 (2.94)	0.80 (0.05)
	Distillation $p_{distill} = 5\%$	25.3 (3.28)	0.81 (0.05)	22.5 (3.13)	0.63 (0.09)	23.9 (3.35)	0.76 (0.07)	25.4 (3.28)	0.81 (0.05)
	Distillation $p_{distill} = 10\%$	25.3 (3.41)	0.80 (0.06)	22.7 (3.40)	0.64 (0.11)	23.9 (3.59)	0.75 (0.08)	25.4 (3.43)	0.80 (0.06)
	Distillation $p_{distill} = 20\%$	25.3 (3.44)	0.80 (0.05)	22.5 (3.12)	0.62 (0.10)	23.7 (3.71)	0.76 (0.07)	25.4 (3.44)	0.80 (0.05)
SIREN $\omega_0 = 15, \omega_i = 15$	Naive	16.6 (2.99)	0.07 (0.04)	16.6 (3.01)	0.07 (0.04)	17.8 (3.44)	0.14 (0.07)	31.7 (4.26)	0.90 (0.07)
	Distillation $p_{distill} = 5\%$	23.7 (3.51)	0.57 (0.08)	24.1 (3.51)	0.60 (0.08)	26.6 (4.07)	0.73 (0.10)	30.3 (3.68)	0.89 (0.06)
	Distillation $p_{distill} = 10\%$	23.9 (3.51)	0.58 (0.08)	24.1 (3.55)	0.60 (0.08)	26.2 (4.20)	0.71 (0.11)	30.3 (3.96)	0.89 (0.07)
	Distillation $p_{distill} = 20\%$	24.0 (3.34)	0.60 (0.07)	24.2 (3.40)	0.61 (0.07)	26.2 (4.11)	0.72 (0.10)	30.3 (3.59)	0.89 (0.06)
FINER $\omega_0 = 5, \omega_i = 5$	Naive	16.8 (3.46)	0.20 (0.06)	16.8 (3.51)	0.20 (0.06)	19.1 (3.21)	0.28 (0.05)	35.3 (3.58)	0.86 (0.05)
	Distillation $p_{distill} = 5\%$	21.0 (2.91)	0.41 (0.07)	21.1 (2.87)	0.41 (0.08)	25.6 (3.18)	0.57 (0.06)	30.9 (2.24)	0.80 (0.04)
	Distillation $p_{distill} = 10\%$	21.0 (2.90)	0.40 (0.07)	21.0 (2.87)	0.40 (0.08)	25.7 (2.89)	0.56 (0.06)	30.9 (1.99)	0.79 (0.04)
	Distillation $p_{distill} = 20\%$	20.9 (3.02)	0.38 (0.05)	20.9 (2.97)	0.37 (0.05)	25.5 (3.04)	0.69 (0.04)	31.0 (1.78)	0.90 (0.02)
DINER L=1	Naive	21.7 (5.17)	0.48 (0.40)	23.8 (8.32)	0.47 (0.49)	27.1 (12.5)	0.48 (0.40)	41.6 (41.5)	0.50 (0.43)
	Distillation $p_{distill} = 5\%$	35.6 (5.48)	0.93 (0.05)	36.8 (6.15)	0.95 (0.04)	48.9 (6.30)	0.99 (0.01)	54.2 (9.60)	0.99 (0.01)
	Distillation $p_{distill} = 10\%$	37.1 (6.73)	0.93 (0.05)	38.8 (5.24)	0.96 (0.03)	51.8 (12.5)	0.99 (0.01)	59.6 (20.9)	0.99 (0.01)
	Distillation $p_{distill} = 20\%$	36.5 (6.00)	0.92 (0.07)	38.6 (5.64)	0.94 (0.05)	47.6 (6.58)	0.98 (0.01)	50.5 (9.88)	0.98 (0.01)

TABLE XII: Imagenet Stabilisation Results

Training Type	Task 1		Task 2		Task 3		Task 4	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Naive	16.7 (5.46)	0.69 (0.23)	16.8 (4.56)	0.70 (0.23)	18.6 (5.79)	0.75 (0.28)	36.3 (12.1)	0.84 (0.34)
Distillation $p_{distill} = 20\%$	34.3 (6.42)	0.93 (0.19)	34.6 (8.29)	0.91 (0.24)	36.5 (8.95)	0.91 (0.25)	38.0 (10.1)	0.92 (0.25)
Stabilised	36.6 (7.51)	0.94 (0.20)	30.4 (8.05)	0.89 (0.24)	29.4 (8.37)	0.88 (0.25)	30.6 (9.18)	0.88 (0.24)

TABLE XIII: ACDC 4D Stabilisation Results. For readability, only the first and last two tasks are included.

Training Type	Task 1		Task 2		Task 11		Task 12	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Naive	21.7 (5.17)	0.48 (0.40)	23.8 (8.32)	0.47 (0.49)	27.1 (12.5)	0.48 (0.40)	41.6 (41.5)	0.50 (0.43)
Distillation $p_{distill} = 20\%$	36.5 (6.00)	0.92 (0.07)	38.6 (5.64)	0.94 (0.05)	47.6 (6.58)	0.98 (0.01)	50.5 (9.88)	0.98 (0.01)
Stabilised	97.2 (55.1)	0.98 (0.04)	86.7 (52.0)	0.91 (0.24)	85.4 (52.1)	0.91 (0.24)	86.7 (52.5)	0.91 (0.24)