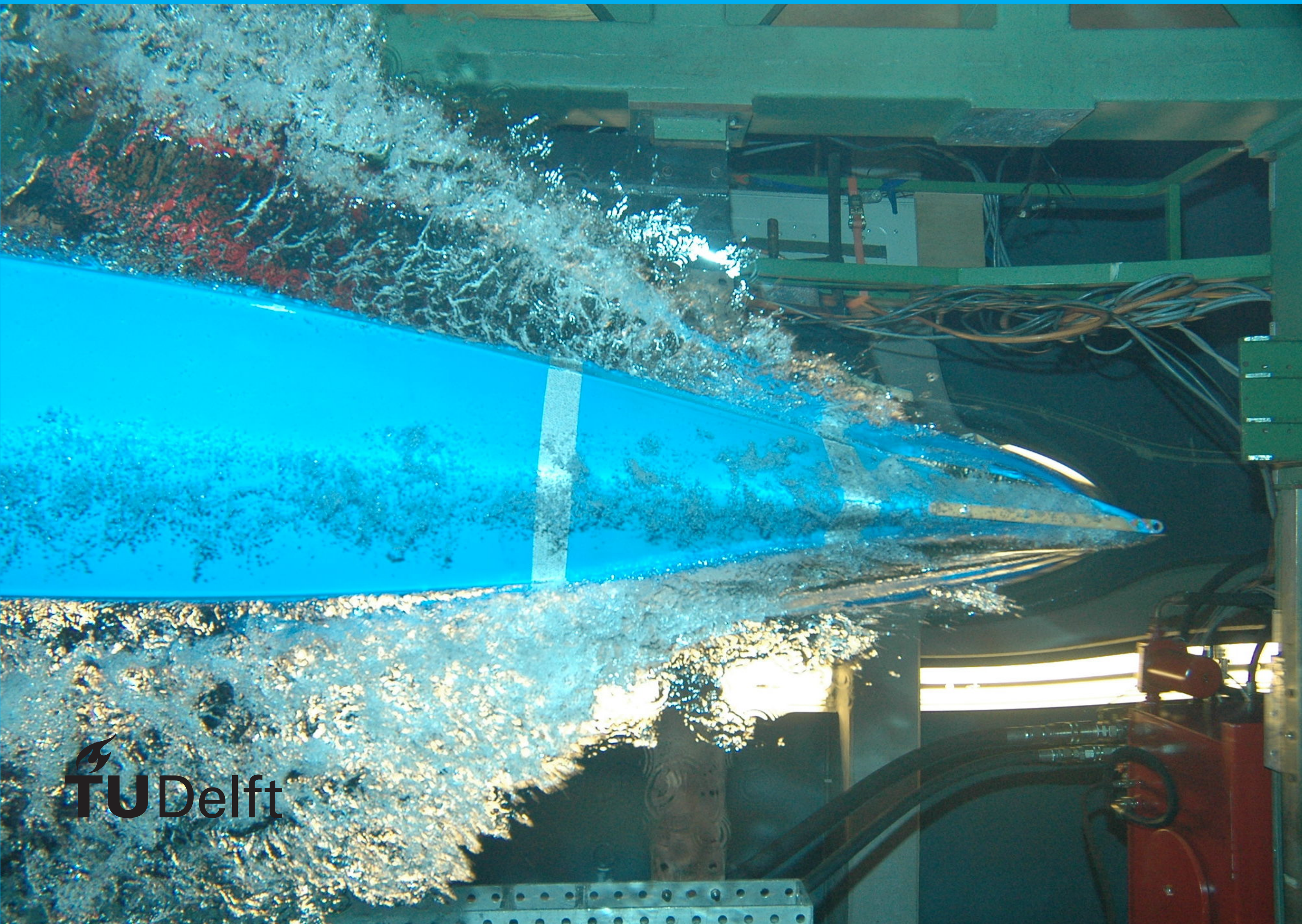
Effect of recurrence on adversarial robustness

Agrim Sharma



Effect of recurrence on adversarial robustness

by

Agrim Sharma

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday August 26, 2022 at 12:00 PM.

Student number:	5353181	
Project duration:	November 26, 2021 – August 26, 2022	
Thesis committee:	Dr. J. van Gemert,	TU Delft, Chair
	Dr. N. Tömen,	TU Delft, Supervisor
	Dr. M. Weinmann,	TU Delft, Graduation Committee

TU Delft

Preface

This report documents the research that I have conducted for my Master's Thesis on Recurrent models and their Effect on Adversarial Robustness. The work shown in the report is a result of working with the Pattern Recognition and Bioinformatics group for the past 9 months, under the supervision of Dr. Nergis Tömen. I would like to thank her for her constant support throughout my thesis, and for the insight she has given me to improve my skills as a researcher. I would also like to thank her for introducing the topic to me, and providing me with the necessary guidance for me to produce work that can live up to her expectations.

I would like to thank Dr. Jan van Gemert for his feedback and perspective, which was a significant help in concentrating the important information. His feedback allowed me to proceed with a strong sense of direction. I would also like to thank Michael Weinmann for giving me his time of day and being a part of my thesis committee. Doing a thesis under the Pattern Recognition and Bioinformatics research group was hugely impactful in improving me as a deep learning researcher, and the resources provided to me continue to help me expand my skillset.

Lastly, I would like to thank my family, my girlfriend, and my friends for providing me with the support and motivation I needed in order to produce high-quality work. I am grateful for all the help I have received from everyone I have interacted with over the past two years, as these experiences have now culminated into my Master's Thesis being a piece of work that I can be proud of.

Agrim Sharma
Delft, August 2022

Contents

1
Scientific Paper

The effect of recurrence on adversarial robustness

Agrim Sharma
Delft University of Technology
The Netherlands
`agrimsharma20@gmail.com`

Nergis Tömen
Delft University of Technology
The Netherlands
`n.tomen@tudelft.nl`

Jan C. van Gemert
Delft University of Technology
The Netherlands
`j.c.vangemert@tudelft.nl`

Abstract

Traditionally, convolutional neural networks are feedforward networks with a deep and complex hierarchy. Conversely, the human brain has a relatively shallow hierarchy with recurrent connections. Replicating this recurrence may allow for shallower and easier to understand computer vision models that may possess characteristics usually attributed to the brain. One of the characteristics we examine in this paper is the effect of recurrence on the robustness of the model against noise. Additionally, we vary the type of noise in order to observe what behaviour is universal and what behaviour is specific to the noise. However, it is crucial to make the distinction that recurrence in the brain is different from that in classical recurrent models. The key difference lies in how information passes through the layers at every step. This gap has been addressed by CORnet, a family of models that use a more biologically compatible hierarchy of recurrence.

In this paper, we take the idea of the biologically compatible recurrent hierarchy and look at the effects of recurrence when applied to a baseline down-scaled Resnet model. We show different settings where recurrence results in an increase in adversarial robustness, and settings where recurrence has the opposite effect. To offer an additional perspective of robustness, we also show how recurrence makes the feature maps of a model more resilient to perturbations. Following these observations, we conclude that biological recurrence allows the model to average out white noise spatio-temporally and we use the observations to support our hypothesis.

1. Introduction

Computer vision has long history of taking inspiration from biological models for human vision [1], [2]. CNNs have had a significant influence on the present state-of-the-art in computer vision [3], showing that drawing analogies between CNNs and biological models of computer vision have been proven to be quite useful in the past [4], [5]. This relationship has been valuable for neuroscience as well, allowing researchers to test decades old hypotheses using CNNs [6]. As such, seeing that CNNs as well were inspired by the study of biological vision [7], it may be fruitful to continue this inclination towards bio-inspired models, further resulting in improvements in computer vision [8], [9].

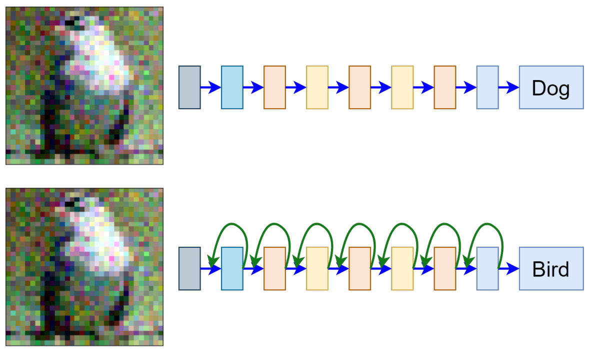


Figure 1. Adding recurrence to feedforward models has been shown to improve the model robustness. In the Figure, a picture of a bird corrupted by Standard Gaussian noise is the input for a feedforward network and a network with backward connections used in Section 4. The feedforward network incorrectly predicts dog with high confidence while the recurrent network is able to correctly predict bird.

There is a considerable overlap between the fundamental principles of CNNs and computational models of biological vision, such as the increasing complexity of features as one looks deeper into the model hierarchy [10]. However, despite this overlap, there are significant differences as well. CNNs lack recurrent connections that are common in the ventral visual stream of primate brains [5], [11]. This opens up CNNs to architectural changes that might help bridge the gap mentioned earlier, leading to increased model-brain parallelism. Additionally, recurrence in Recurrent Networks is different from recurrence in the brain in terms of the way information passes through the model over time [12]. This means classical recurrent models cannot simulate visual processing the way that is done by the primate brain. This difference between standard recurrence and biological recurrence is explained in Section 2.1.

To look at the effects of adding recurrence, we looked at the impact recurrence has on the robustness of a model against adversarial noise in the input. Adding recurrent connections is thought to bring contextual information to early visual areas to fill in missing details, aiding in robustness [11], [13]. Similarly, top-down processing performed by the recurrent connections is shown to improve sensory representations in the brain [14]. Recurrent connections are a vital part of the visual system that aid the robustness of object recognition in humans [15], [16]. Conditions where recurrence does not come into play, such as time-limited observations, result in humans making similar mistakes as computational models [17]. In this paper, we use different types of attacks to test this theory, and propose our own alternative explanation that recurrent connections carry out spatio-temporal averaging which helps combat noise. We show conditions that result in recurrence being detrimental to robustness, and provide evidence for why our explanation is consistent with the results. Lastly, we consider robustness with respect to the stability of validation accuracy with increasing noise, and also with respect to the sensitivity of the feature maps to noise. In both cases, we find that the type of noise plays a major role in whether the model shows improved robustness.

Our main contributions are:

- Showed that adding recurrence to a model increases the robustness (with respect to validation accuracy) against white noise introduced such as Standard Normal Gaussian Noise at test time. This increased robustness holds true with and without adversarial training.
- Showed that degree of recurrence has a significant impact on robustness, and that there is a sweet spot in the amount of recurrence which maximizes robustness.
- Showed that the robustness due to recurrence does not apply for correlated noise and similar perturbations that are not white noise.
- Showed that neural activations of recurrent models are more likely to recover from small perturbations to the input.

2. Related Work

2.1. Biologically Recurrent Networks

The paper by Kubilius et al [12] introduces the concept of biological recurrence and how it can be used in a CNN to create models that are biologically more analogous to the brain. In their work, the researchers introduce a family of models known as CORnet that use biological recurrence to process information. We have adopted their work to construct a biologically recurrent resnet [18], which is then used for the experiments in Section 4.

The order of computation is crucial when a network needs to pass data through its constituent blocks multiple times. In traditional recurrent networks, one time step is characterized by information passing from the input all the way to the output [19]. However, in order to have a biologically consistent form of recurrence, the information is passed through in a stepwise manner [12]. This would mean that each forward pass would consist of multiple time steps, where each time step sees the output of a layer feeding back into itself, and also passing only to the very next layer. For example, in Figure 3, at t=0, only the 3x3 conv block will process information and pass it to Residual Block 1. At t=1, Residual Block 1 will process that input, and the 3x3 conv block will update its state and begin processing the next input. When all time steps have taken place, one forward pass is complete. This mechanism can be seen in Figure 2.

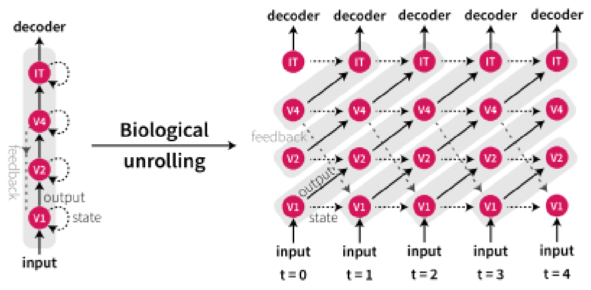


Figure 2. A biologically recurrent network when unrolled shows the stepwise manner in which information passes from one layer to another. The deeper layer only receives its input in the next time step if the previous layer received an input in the previous time step. This figure has been referenced from the paper by Kubilius et al [12].

2.2. Adversarial Robustness

CNNs lack the human brain's robustness against perturbations [20], [21], which is an important shortcoming that has receives a lot of attention due to its implications for the increased performance and safety of computer vision applications [22]. Figures 5 and 10 shows the types of noise used for the adversarial attacks. One example of a successful adversarial attack during training showed how minor noise (with 1% of the intensity of the noise shown in the figure) is able to completely change the prediction from plane to deer, with the network being very confident in its incorrect prediction. One reason for this observation is due to the fact that CNNs have a tendency to have high texture bias while humans have a tendency to have a high shape bias [23]. Due to this, any minor perturbations can change the texture enough to change the model's predictions.

The most popular method of adding robustness to a model is to include examples of adversarially attacked images in the training dataset [24]. Doing so allows for the model to incorporate adversarial examples into their feature maps, making the model more robust against attacks of intensities that it has not experienced before. However, this method is highly specific as we can only train on known attacks, making the network susceptible to newer or different types of attacks. Recurrent connections in the model are hypothesized to aid robustness by bringing contexual information from deeper layers in order to fill in missing or corrupted information in the present layer [11]. The top-down feedback has a similar effect as increasing the receptive field of a layer, thus incorporating more contextual information to the neuron responses and helping the model resolve local ambiguity [13].

Section 4 is dedicated to addressing this further, and presenting situations where the hypothesis doesn't hold true, and why that may be the case.

3. Method

3.1. Recurrent Resnet

As a baseline, we used a scaled down version of Resnet18, shown in Figure 3. The standard resnet model only had the blue feedforward connections. The green connections were added to make the model recurrent based on the work done in the paper introducing CORnet [12]. The resulting models have the same number of parameters, as shown in Table 1.

Additionally, we used a coarse to fine recurrent model, where the input in the first time step is heavily blurred, getting finer with each time step, with the final image being the original, unchanged image. Figure 4 shows what the input looks like every time step. This was done in order to assess whether the model will learn better features if it receives the input image at multiple resolutions over time, in comparison to the standard recurrent model, that receives the original image for every time step. The coarse to fine mechanism may also help the model have a higher shape bias due to the gradual deblurring of the input image [25], but this requires further analysis.

3.2. Adversarial Attacks

In this paper, we use two different blackbox attacks [26], [27] throughout all the experiments. The attack is blackbox as the process of generating the noise does not have any information about the target network [28]. Standard Normal Gaussian Noise was used as it is the basis of a major part of the work done in adversarial robustness [24], [29]. Additionally, it allows us to see the effects on the accuracy of the model when the noise is uncorrelated white noise. Figure 5 shows a 32x32 patch of Gaussian Noise, like the ones used

Model Name	Accuracy (%)	Number of Parameters
Resnet	89.0 $\pm$0.2	554634
Recurrent Resnet	88.8 $\pm$0.3	554634
Recurrent Resnet CtF	89.5 $\pm$0.2	554634

Table 1. Adding recurrent connections does not affect the number of parameters, but it has a noticeable effect on the Accuracy.

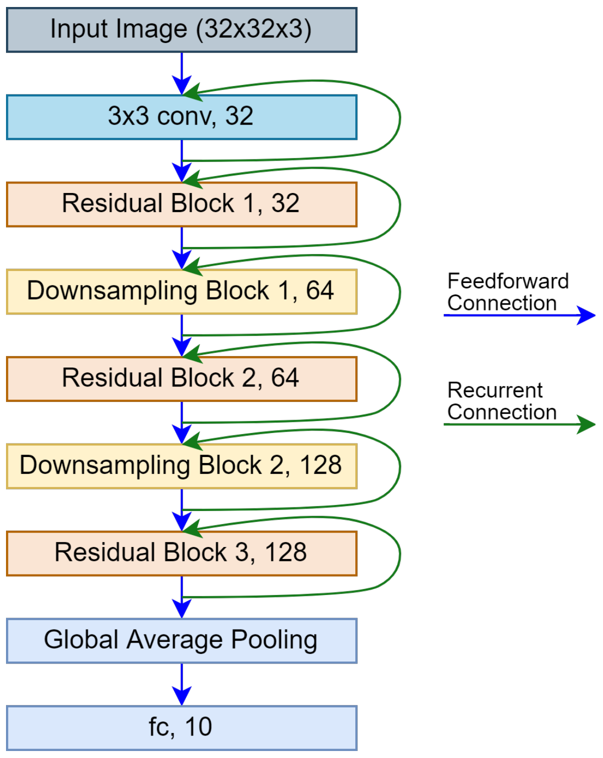


Figure 3. Architecture for the models used. The baseline model only consists of the blue feedforward connections, while the recurrent model is the same as the baseline models but with the recurrent connections shown as green lines.

in Section 4.

On the other hand, in order to assess how the model responds to perturbations other than white noise, we used correlated noise, shown in Figure 10. This noise was generated as shown in algorithm 1.

To perform the adversarial training, we included adversarial examples in the training data. The transforms that the examples had have been summarized below:

- Experiment 1: Standard Gaussian noise with up to 1% of the intensity of the original image was added to the adversarial examples during training time.

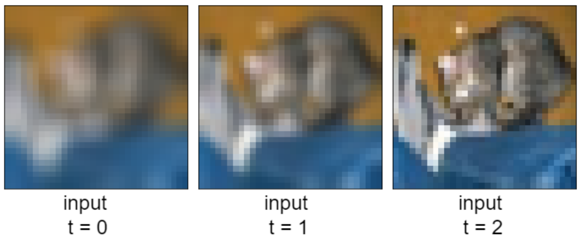


Figure 4. For 3 time steps per forward pass, the coarse to fine model receives increasingly fine inputs every time step. The effect is exaggerated in the image for visualization purposes.

Algorithm 1 Algorithm to generate correlated noise

```
$Func \leftarrow$ Define a function, $f(x) = (\frac{1}{f})^x$ in Frequency Space
$Rand \leftarrow$ Define an uncorrelated Random Function in Frequency Space
$Complex\ Noise \leftarrow$ Product of $Func$ and $Rand$
$Correlated\ Noise \leftarrow$ Convert $Complex\ Noise$ to real space
```

- Experiment 3: Standard Gaussian noise with exactly 1% of the intensity of the original image was added as a rectangular patch centered on the image during training time. The patch ranged between 10% to 90% the size of the original image.
- Experiment 4: Correlated noise with up to 5% of the intensity of the image was added to the adversarial examples during training time.

Testing a recurrent CNN with various adversarial attacks allows us to confirm whether recurrence makes a model universally robust, or if the robustness is specific to the types of perturbations. In addition to that, we look at the feature maps produced by the model before and after adversarial attacks to help us visualize how the extracted features are affected.

4. Experiments

The models used in the experiments have been explained in Section 3.1.

4.1. Experiment 1: Gaussian Noise

As a baseline, we tested how biological recurrence affects robustness against Gaussian Noise. In order to do this, we used three models- resnet, resnet recurrent, and resnet recurrent (coarse to fine). The models were trained on cifar10 [30] once with standard training, and once with adversarial training. During adversarial training, the Gaussian noise was weighted between 0 and 0.01 (1% of the original image intensity). The noise is chosen to be imperceptibly small since the model should be able to generalize to examples with more severe noise during test time in order for the adversarial training to be useful. Figure 5 shows how an image is affected by the noise.

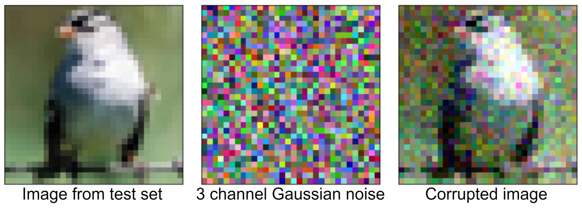


Figure 5. Example of an image of a bird affected by Gaussian Noise. The effect has been exaggerated for visualization purposes, but during training time, the noise weight is 1% of the example image.

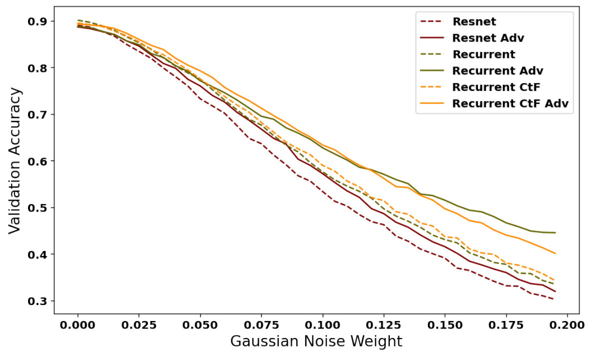


Figure 6. The validation accuracy vs the Gaussian noise weight. All models with suffixed with 'Adv'are models that have undergone adversarial training.

To test the robustness, the models mentioned above were adversarially attacked with noise of increasing weight at test time. Figure 6 show the trend that the models follow. We find that the recurrent models are more robust in comparison to the baseline resnet model even when none of the models are adversarially trained. This difference becomes greater when the models are all trained with adversarial samples, showing a greater ability to generalize, even at noise weights larger than those the models were trained on. Additionally, we see that in Figure 6, for noise weight below 0.1, the coarse to fine recurrent model outperforms the standard recurrent model. The increasing sharpness of the image with each time step shows an additional method for improving robustness of a model, but further research is required to make any claims.

The models had increased robustness due to recurrence even when they were not optimized for robustness, like in the case of models with no adversarial training. This hints at a mechanism of recurrence that makes a model inherently more robust. This line of enquiry is explained in further detail in the next section.

4.2. Experiment 2: Degree of Recurrence

Our hypothesis for the increase in the robustness of recurrent models is that these models are able to average out the Gaussian noise, since uncorrelated noise can be easily averaged out over time. Similarly, the CtF model further averages the noise out over space, leading to added robustness. To test whether this mechanism aids robustness monotonously, we repeated the previous experiment with models of varying degree of recurrence.

Every time step, a layer receives feedback from itself along with input from the previous layer. Once every time step has passed, one forward pass is completed. By changing the number of time steps in one forward pass, we can change how recurrent the model is. This allows us to analyze the effect that the degree of recurrence has on the robustness of the model. Figure 7 shows this relation.

We find that while recurrence increased the robustness to Gaussian Noise in Experiment 1, it does not always have that effect. If the model requires too many, or too little time steps to complete the forward pass, it ends up performing worse than the baseline resnet. The optimal number of time steps for this experiment can be seen to be around 5.

The papers by Yan et al [13] and Choi et al [11] state that recurrence allows a layer to receive contextual information from deeper layers in order to fill in missing information. Figure 7 shows that having a model that is not recurrent enough may provide an insufficient amount of information from deeper layers to maintain a relatively high validation accuracy. Consequently, a model that is too recurrent would also result in a drop in accuracy as the original image would have decreasing contribution relative to the information from all the other time steps. This would result in the the number of time steps being a hyper-parameter in need of tuning specific to the problem.

Figure 7 could also help explain why the coarse to fine recurrent model in Figure 5 drops below the standard recurrent model after noise weight 0.1. The additional blurring from the coarse to fine mechanism may be analogous to the averaging mechanism of a more recurrent model. This would result in the original image, which is only available to the coarse to fine model for the final time step, having too little contribution towards deciding its class.

4.3. Experiment 3: Local Noise

We saw from the previous experiments that recurrent models show a tendency to be more robust to adversarial at-

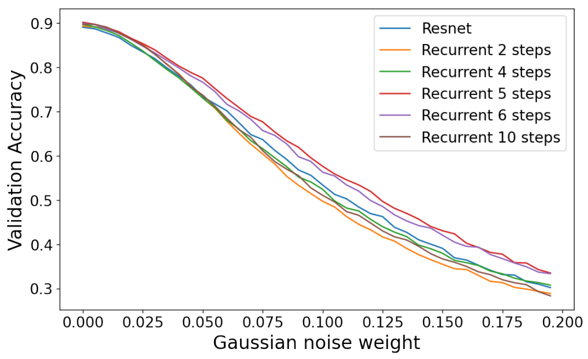


Figure 7. The validation accuracy vs weight of the Gaussian noise. Each model has undergone the same training, but the number of time steps per forward pass in different, making some models more recurrent than others.

tacks even without any adversarial training, provided their degree of recurrence is tuned correctly for the task. However, the noise in the experiments affected the entire image. Studies have shown that even a highly local attack like a patch [31], [32], or even a single pixel [33] is enough to fool the model. In this experiment, we aimed to show how validation accuracy is affected during test time as noise varies from local to global. If our hypothesis is true, we should see the recurrent models perform better than the baseline as the noise covers a larger portion of the image due to the spatio-temporal averaging providing robustness against the uncorrelated noise. As the noise spreads, the recurrent models will be in an increasingly advantageous condition.

We use the same models from Experiment 1, but the noise is changed to a patch of Gaussian noise centered on the image. The size of the patch of noise is variable in order to cover anywhere between 0% and 100% of the image in order to assess the accuracy at varying levels of locality. While training, the weight of the noise was fixed at 1% of the image intensity, and it covered between 10% to 90% of the image. Figure 8 shows what the image looks like when adversarially attacked.

In order to assess how the accuracy is affected, during test time, the noise weight was kept constant at 5% of image intensity and the patch of noise was varied to cover the entire image. We can see this in Figure 9. We find that as the area covered by noise increases, the recurrent models are able to maintain their accuracy better than the baseline resnet models.

The increasing difference once again points at an advantage provided by recurrence that helps maintain the accuracy while the model is under attack. In addition to providing a layer contextual information from deeper layers [13], we demonstrate that the recurrent model is able to average out white noise over multiple time steps, enabling it to extract features more effectively in the presence of Gaussian noise. The noise used has zero mean, which would mean that over enough time steps, the noise becomes ineffective.

This could also explain why the coarse to fine model outperforms the standard recurrent model, as the additional averaging from the blurry images could provide more information about the features of the image.

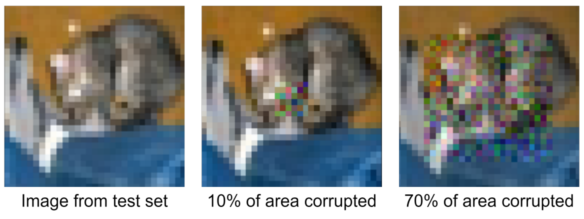


Figure 8. Example of an image of a cat affected by Local Noise. The area of the image corrupted can be varied in order to affect how local or global the noise is.

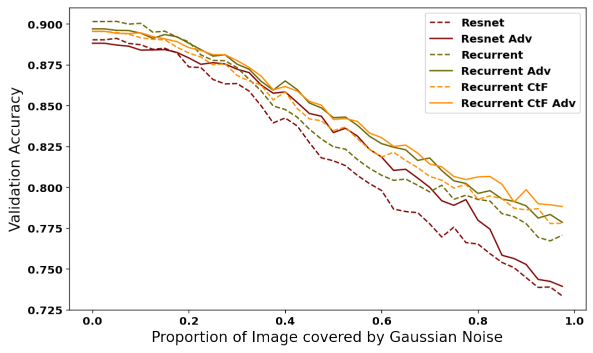


Figure 9. The validation accuracy vs the proportion of image covered by Gaussian noise. As a larger portion of the image gets corrupted, the validation accuracy begins to decrease.

4.4. Experiment 4: Correlated Noise

The previous experiments showed an increase in robustness when white noise is used. Based on our hypothesis that recurrent models are able to average out noise spatio-temporally, if we used noise that does not average to zero over time, we would not see the added robustness as we did earlier. To verify this, we use correlated noise, generated using algorithm 1. While training, the noise varied from 0% to 5% of the original image. Figure 10 shows the effect of correlated noise on an image.

Testing the robustness required assessing the validation accuracy at differing noise weights. Figure 11 shows the result of attacking the models with correlated noise. We find that recurrent models no longer have an advantage over the baseline resnet models. In fact, they perform significantly worse under these conditions.

This shows us that when we use a non-normally distributed noise such as correlated noise, the recurrent models, both adversarially trained and untrained, perform worse than their baseline resnet counterparts. These results indicate that in the presence of perturbations that do not average to zero over time, recurrence starts to be a disadvantage. This further supports the hypothesis that averaging over time is a mechanism used by biologically recurrent models which as a consequence affects robustness differently under different conditions.

This could also explain why the coarse to fine models perform the worst, as the additional averaging from the blurry images would put the model at a bigger disadvantage when attacked by correlated noise.

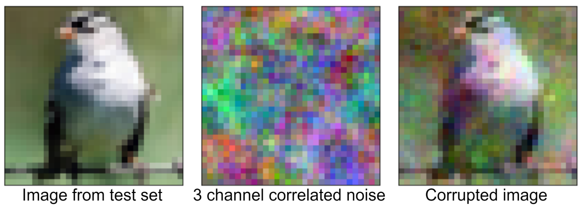


Figure 10. Example of an image of a bird affected by Correlated Noise. The noise is a lot less grainy than Gaussian nose as the correlated nature of the noise allows for smoother transitions between regions of the image.

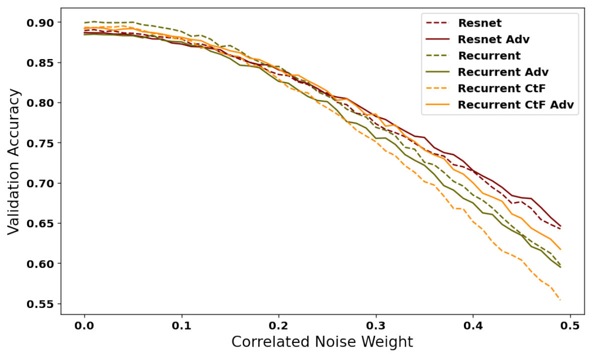


Figure 11. The validation accuracy vs the weight of the correlated noise. We can see that the recurrent models no longer outperform the resnet model for high noise weights.

4.5. Experiment 5: Degree of Correlation

Once it was established that recurrent models perform poorly under correlated noise, we fixed the noise weight at 10% of image intensity and varied the degree of correlation of the noise at test time to visualize the response of the models under a more generalized setting. The degree can be varied by changing the value of x in Algorithm 1. When x=0, the noise reduces to Gaussian noise, at x=-1, the correlated noise is the same as the noise in Experiment 4, and as x reduces, the noise tends towards being more correlated. Figure 12 shows this response.

We find that on the left side of degree of correlation at -1, there is a rapid drop in the accuracy of recurrent models, while the baseline resnet models are only affected with a significantly higher amount of correlation. Consequently, to the right of degree of correlation at -1, we see the recurrent models performing better than the baseline resnet models as the degree of correlation decreases and the noise approaches Gaussian.

These results are consistent with the previous experiments as at noise weight 10%, Figure 6 shows the validation accuracy of the models to be between 50% to 60%, which is what we see in Figure 12 at degree of correlation at 0. Similarly, Figure 11 shows the validation accuracy at 90% at 10% noise weight, which is what we see in Figure 12 at degree of correlation at -1.

These findings provide additional evidence for the existence of an averaging mechanism in biologically recurrent models which also explains the reversal of the order of the most robust models as the noise approaches a more uncorrelated setting.

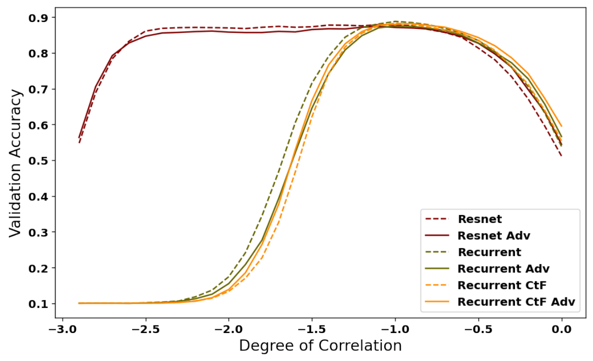


Figure 12. The validation accuracy vs the Degree of Correlation. The recurrent models lose accuracy drastically as the degree of correlation increases while the resnet models are able to maintain their accuracy much better. Notice how the order of most robust model flips as the noise approaches Gaussian noise.

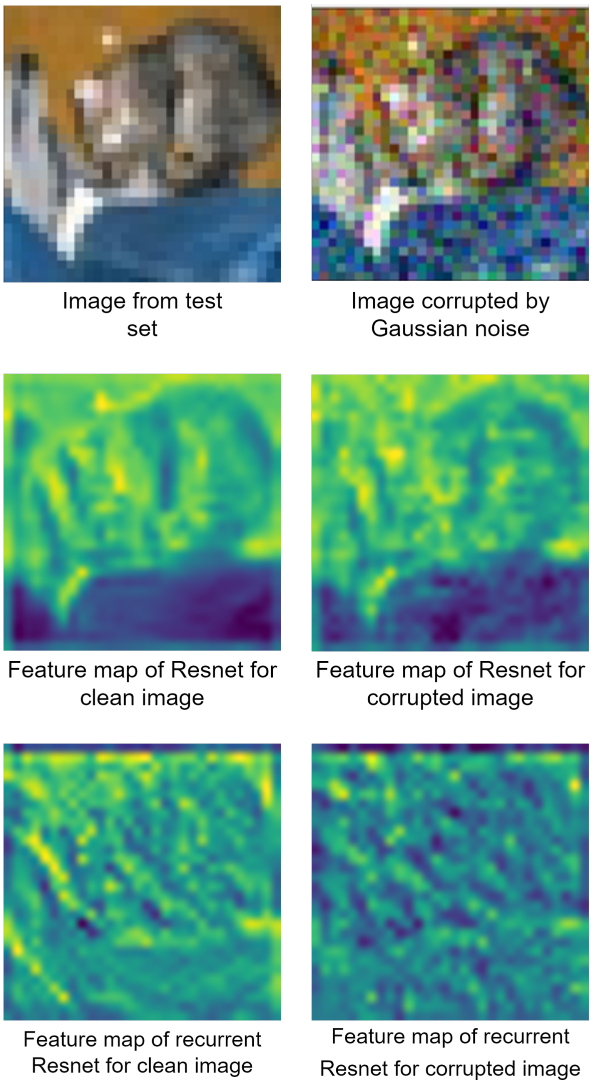


Image from test set
Image corrupted by Gaussian noise
Feature map of Resnet for clean image
Feature map of Resnet for corrupted image
Feature map of recurrent Resnet for clean image
Feature map of recurrent Resnet for corrupted image

Figure 13. Feature Maps of an individual channel in Block 1 of both resnet, and recurrent resnet models. The images on the left have the clean image as input, and the images on the right have the corrupted image as input.

4.6. Experiment 6: Effect of noise on Extracted Features

To visualize the effect of recurrence on the feature maps produced by the models, we show in Figure 13 the feature maps of one channel of one layer of the models. The input image is then corrupted and the feature maps are produced again to visualize the perturbations caused in the feature maps due to the perturbations in the input image.

We then calculate the Mean Squared Error between a feature map with a clean input image, and a feature map from the same channel with a corrupted input image. This gives us a metric for how much the feature maps are affected by noise, with low Mean Squared Errors indicating more robustness in terms of preserving the detail in the feature maps. Doing this calculation on 1000 images provides us with Figure 14, which shows how some layers preserve feature maps better than other layers.

A general trend that can be observed in Figure 14 is that, at all times, the adversarially trained models have lower Mean Squared Error than their counterparts with standard training. This shows an increase in robustness with a different perspective from that in Figures like Figure 5, where robustness is more literally defined as the model's ability to prevent accuracy loss.

We see in Figure 14 that the recurrent models have a higher Mean Squared Error than the baseline resnet in the earlier layers, and a lower Mean Squared Error in the later layers. Since later convolution layers have a larger number of channels than earlier layers, having lower Mean Squared Error in the later layers results in a lower net loss of information. We can calculate this taking the sum of the mean squared error over every layer, weighted by the number of channels in the layer.

As the noise used for this experiment was Gaussian, we can expect after seeing all the previous results that the better the ability to average out the noise is, the more robust the model will be. Using the weighted sum metric, this is exactly what we see. We find the coarse to fine recurrent model to be the most robust, the baseline resnet model to be the least robust, with the standard recurrent model in the middle.

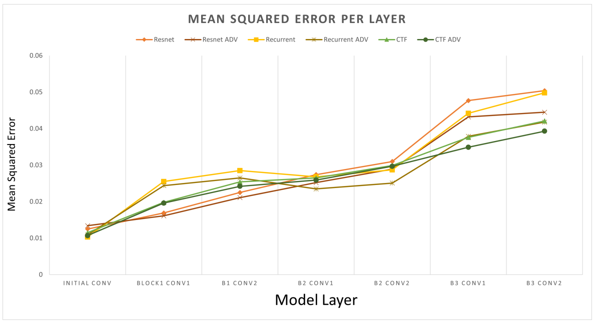


Figure 14. Mean Squared Error between clean and corrupted feature maps over all the layers of the models. Resnet models show a roughly linear trend until the later layers, where the error increases drastically. Meanwhile, recurrent models have higher error in the earlier layers, and relatively lower error in the later layers.

4.7. Experiment 7: Real Life Dataset

Lastly, this experiment shows the performance of recurrent models in a real world setting. We used a dataset of rocks to test whether recurrence is able to find any practical uses. The rock dataset was chosen as the rocks contain texture and impurities, which act as both features and as noise. The texture could contain information about type of rock, but the impurities could prevent the model from making the correct prediction. This dataset would require a trade-off between ignoring some patterns while keeping other, useful patterns. For this reason, we hypothesized the recurrent model would have a slight advantage due to the reasons discussed earlier. Figure 15 shows an example from each of the seven classes in the dataset.

We can see the results of the training in Table 2. We see that the standard recurrent model outperforms the baseline resnet model. Additionally, we see that the coarse to

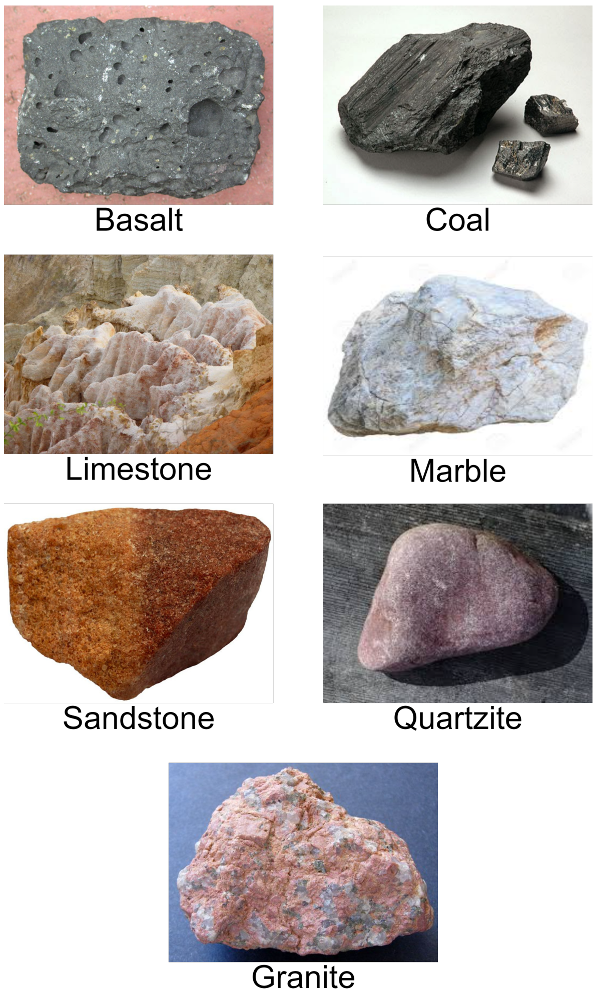


Figure 15. An example from each class of rocks in the dataset. We see the rocks have plenty of texture but also contain impurities.

fine model performs the worst. One possible explanation using our hypothesis is, as we stated above, the trade-off between ignoring and keeping texture information. The recurrent connections in the model would give it the spatio-temporal averaging ability to ignore the real world noise in the data, but the subtle effects of this averaging could allow the model to still preserve textures that correlate directly to an increase in performance. This effect would result in the right sweet spot for being able to maximize the information extracted from the dataset.

On the other hand, the poor performance of the coarse to fine model can be explained by the fact that the textures indicative of the class of rock are important features that are blurred for the majority of the time steps in the coarse to fine model. This blurring would cause a loss of too much information from the model, and would get rid of both the unneeded noise, but the important textures, resulting in a

Model Name	Accuracy (%)	Number of Parameters
Resnet	60.0 ± 0.25	554634
Recurrent Resnet	61.4 ± 0.4	554634
Recurrent Resnet CtF	55.7 ± 0.3	554634

Table 2. Adding recurrence shows an improvement in accuracy for the standard recurrent model. However, for the coarse to fine model, the performance drops considerably.

significant drop in accuracy, which we see in Table 2.

5. Conclusion

The experiments show that biological recurrence could have a positive influence on robustness in specific situations.

We also see that this robustness does not hold against correlated noise, and is detrimental to the model's performance. This fact is true for correlated noise at any degree of correlation, which hints at a the possibility of an alternate mechanism for robustness. Instead of filling in missing contextual cues from later layers, biological recurrence might be averaging over white noise in order to extract features more effectively. We even see in Experiment 7 that this mechanism can have practical uses.

Experiment 6 provides a graphic for why recurrent networks are more robust, showing the effect of input perturbations on the internally observable activations. According to Figure 14, in comparison to resnet, earlier layers of recurrent models are more prone to perturbations, while later layers are more robust. This has the overall effect of increased robustness due to a larger proportion of channels being in the later layers.

6. Discussion

While the evidence above suggests that recurrence can be largely beneficial for the future of reliable and noise resistant computer vision models, adding recurrence to a model comes with its downsides as well. Experiment 2 shows us that the degree of recurrence is an important factor in determining whether the model will benefit from robustness as the effect is lost when the model is not recurrent enough, and the contribution of the original image is diminished if the recurrence is too high. This would result in the number of time steps for the recurrent connections being another hyperparameter that needs to be tuned to optimize performance.

Additionally, performing recurrent computations results in the model taking more time to train, even if the number of parameters in the feedforward and recurrent models is the same. From our experiments, we observed that recurrent models take on average 3x more time than their feedforward counterparts due to the stepwise nature of the computations. Whether this cost is acceptable for the added effects of recurrence depends on the application, but in situations where safety is involved, recurrence may be quite useful.

However, more work is needed to be confident in these claims. Varying the degree of correlation for adversarial training might show interesting results as the model would be trained on a wider range of a family of noise. Furthermore, using other types of noise, and real world examples of perturbed data would shed more light on the effects of recurrence. We tried using Perlin noise for similar experiments as the ones in Section 4, but the results were not very informative. This data along with other experiments can be found in the Appendix.

References

[1] D. D. Cox and T. Dean, "Neural networks and neuroscience-inspired computer vision," *Current Biology*, vol. 24, no. 18, R921–R929, 2014, ISSN: 18790445. DOI: 10.1016/j.cub.2014.08.026.

[2] K. Han, H. Wen, Y. Zhang, D. Fu, E. Culurciello, and Z. Liu, "Deep predictive coding network with local recurrent processing for object recognition," *Advances in Neural Information Processing Systems*, vol. 2018-Decem, pp. 9201–9213, 2018, ISSN: 10495258.

[3] T. J. Sejnowski, "The unreasonable effectiveness of deep learning in artificial intelligence," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 117, no. 48, pp. 30 033–30 038, 2020, ISSN: 10916490. DOI: 10.1073/pnas.1907373117.

[4] T. C. Kietzmann, M. R. C. Cognition, and B. S. Unit, "Deep Neural Networks in Computational Neuroscience Explaining Brain Information Processing Requires Complex , Task-," *Oxford Research Encyclopaedia of Neuroscience*, no. January, pp. 1–29, 2021.

[5] R. Rajalingham, E. B. Issa, P. Bashivan, K. Kar, K. Schmidt, and J. J. DiCarlo, "Large-scale, high-resolution comparison of the core visual object recognition behavior of humans, monkeys, and state-of-the-art deep artificial neural networks," *Journal of Neuroscience*, vol. 38, no. 33, pp. 7255–7269, 2018, ISSN: 15292401. DOI: 10.1523/JNEUROSCI.0388-18.2018.

[6] J. Lindsey, S. A. Ocko, S. Ganguli, and S. Deny, "A unified theory of early visual representations from retina to cortex through anatomically constrained deep cnNs," *7th International Conference on Learning Representations, ICLR 2019*, pp. 1–17, 2019.

[7] G. W. Lindsay, "Convolutional neural networks as a model of the visual system: Past, present, and future," *Journal of Cognitive Neuroscience*, vol. 33, no. 10, pp. 2017–2031, 2021, ISSN: 15308898. DOI: 10.1162/jocn{_}a{_}01544.

[8] J. Dapello, T. Marques, M. Schrimpf, F. Geiger, D. D. Cox, and J. J. DiCarlo, "Simulating a primary visual cortex at the front of CNNs improves robustness to image perturbations," *Advances in Neural Information Processing Systems*, vol. 2020-Decem, 2020, ISSN: 10495258.

[9] J. Kim, O. Sangjun, Y. Kim, and M. Lee, "Convolutional Neural Network with Biologically Inspired Retinal Structure," *Procedia Computer Science*, vol. 88, pp. 145–154, 2016, ISSN: 18770509. DOI: 10.1016/j.procs.2016.07.418.

[10] D. L. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo, "Performance-optimized hierarchical models predict neural responses in higher visual cortex," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, no. 23, pp. 8619–8624, 2014, ISSN: 10916490. DOI: 10.1073/pnas.1403112111.

[11] M. Choi, Y. Zhang, K. Han, X. Wang, and Z. Liu, "Human Eyes Inspired Recurrent Neural Networks are More Robust Against Adversarial Noises," pp. 1–16, 2022. [Online]. Available: http://arxiv.org/abs/2206.07282.

[12] J. Kubilius, M. Schrimpf, A. Nayebi, D. Bear, D. L. Yamins, and J. J. DiCarlo, "CORnet: Modeling the Neural Mechanisms of Core Object Recognition," *bioRxiv*, no. September, 2018, ISSN: 2692-8205. DOI: 10.1101/408385.

[13] S. Yan, X. Fang, B. Xiao, H. Rockwell, Y. Zhang, and T. S. Lee, "Recurrent Feedback Improves Feedforward Representations in Deep Neural Networks," pp. 1–10, 2019. [Online]. Available: http://arxiv.org/abs/1912.10489.

[14] P. Kok, J. F. Jehee, and F. P. de Lange, "Less Is More: Expectation Sharpens Representations in the Primary Visual Cortex," *Neuron*, vol. 75, no. 2, pp. 265–270, 2012, ISSN: 08966273. DOI: 10.1016/j.neuron.2012.04.034.

[15] K. Kar, J. Kubilius, K. Schmidt, E. B. Issa, and J. J. DiCarlo, "Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior," *Nature Neuroscience*, vol. 22, no. 6, pp. 974–983, 2019, ISSN: 15461726. DOI: 10.1038/s41593-019-0392-5.

[16] E. Kim, J. Rego, Y. Watkins, and G. T. Kenyon, "Modeling biological immunity to adversarial examples," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4665–4674, 2020, ISSN: 10636919. DOI: 10.1109/CVPR42600.2020.00472.

[17] G. F. Elsayed, N. Papernot, S. Shankar, *et al.*, "Adversarial examples that fool both computer vision and time-limited humans," *Advances in Neural Information Processing Systems*, vol. 2018-Decem, no. NeurIPS, pp. 3910–3920, 2018, ISSN: 10495258.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016, ISSN: 10636919. DOI: 10.1109/CVPR.2016.90.

[19] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–11, 2015.

[21] R. Geirhos, D. H. J. Janssen, H. H. Schütt, J. Rauber, M. Bethge, and F. A. Wichmann, "Comparing deep neural networks against humans: object recognition when the signal gets weaker," 2017. [Online]. Available: http://arxiv.org/abs/1706.06969.

[22] W. Ruan, X. Yi, and X. Huang, *Adversarial Robustness of Deep Learning: Theory, Algorithms, and Applications*, 1. Association for Computing Machinery, 2021, vol. 1, pp. 4866–4869, ISBN: 9781450384469. DOI: 10.1145/3459637.3482029.

[23] R. Geirhos, C. Michaelis, F. A. Wichmann, P. Rubisch, M. Bethge, and W. Brendel, "Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," *7th International Conference on Learning Representations, ICLR 2019*, no. c, pp. 1–22, 2019.

[24] B. Li, C. Chen, W. Wang, and L. Carin, "Certified adversarial robustness with additive noise," *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019, ISSN: 10495258.

[25] S. Yoshihara, T. Fukiage, H. Information, N. Telegraph, and T. Corporation, "DO TRAINING WITH BLURRED IMAGES MAKE CONVOLUTIONAL NEURAL NETWORKS CLOSER TO HUMANS CONCERNING OBJECT RECOGNITION PERFORMANCE AND INTERNAL REPRESENTATIONS ?," 2022.

[26] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, no. 2, pp. 1–24, 2017.

[27] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," *ASIA CCS 2017 - Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, pp. 506–519, 2017. DOI: 10.1145/3052973.3053009.

[28] H. Masuda, T. Nakai, K. Yoshida, T. Kubota, M. Shiozaki, and T. Fujino, "Black-Box Adversarial Attack against Deep Neural Network Classifier Utilizing Quantized Probability Output," *Journal of Signal Processing*, vol. 24, no. 4, pp. 145–148, 2020, ISSN: 1342-6230. DOI: 10.2299/jsp.24.145.

[29] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–12, 2018.

[30] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, pp. 1–9, 2010. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Convolutional+Deep+Belief+Networks+on+CIFAR-10#0.

[31] K. Eykholt, I. Evtimov, E. Fernandes, *et al.*, "Robust Physical-World Attacks on Deep Learning Models," 2017. [Online]. Available: http://arxiv.org/abs/1707.08945.

[32] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial Patch," no. Nips, 2017. [Online]. Available: http://arxiv.org/abs/1712.09665.

[33] J. Su, D. V. Vargas, and K. Sakurai, "One Pixel Attack for Fooling Deep Neural Networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019, ISSN: 19410026. DOI: 10.1109/TEVC.2019.2890858.

2
Basics of Deep Learning

Deep Learning is a subset of a larger domain called Machine Learning. Deep Learning models such as Convolutional Neural Networks (CNNs) typically consist of multiple layers, and the features extracted by the layers get increasingly complex deeper into the layers [1]. Traditionally, in computer vision, features had to be manually selected for a particular dataset for the model to have success [2]. However, deep learning models are able to learn the features automatically, giving them a high level of flexibility [3]. Deep Learning has since exploded in popularity and CNNs are used in a wide variety of areas like object detection [4], object classification [5], and action detection [6]. In this section, we will explain CNNs in further detail.

2.1. Convolutional Neural Networks

CNNs are a class of Neural Networks that are particularly useful in processing image and video data [7]. In the paper, the CNNs were trained on image data. When an image is given to a CNN as input, the image is represented as a stack of matrices, one for each channel for red, green, and blue. Each matrix element corresponds to the intensity of the pixel of that channel. This stack of matrices is what the convolution layer receives, where the features are extracted from the data[8]. The general structure of a CNN block can be seen in Figure 2.1.

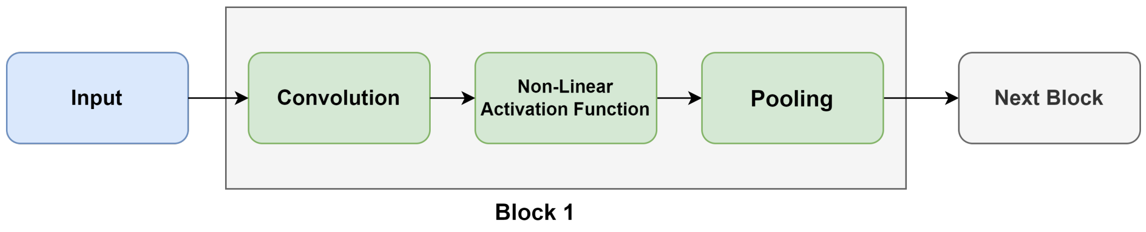


Figure 2.1: The input passes through the convolution layer after which the activation layer activates the neurons with a high response. These neurons generate the feature map, that then passes through the pooling layer in order to be downscaled.

2.1.1. Convolution Layer

A Convolution layer consists of several filters that perform 2D convolution on the input images, across all the image channels, each generating a feature map. These feature maps are then stacked back to back to create another image with N channels where N is the number of filters [9]. This new image is then used as the input for the next layer, increasing the complexity and flexibility of the features that are detected by the CNN overall. In the paper, we have used a Resnet to produce the feature maps. Figure 2.2 shows how convolution works for a single filter, and the formal definition of convolution can be found in the equation below:

$$X_i^{(l)} = b_i^{(l)} + \sum_{j=1}^{D^{(l-1)}} K_{i,j}^{(l)} * X_j^{(l-1)} \tag{2.1}$$

Where $X_i^{(l)}$ is the i^{th} feature map of layer l produced as the output of the previous layer. $b_i^{(l)}$ is the bias, $K_{i,j}^{(l)}$ is the filter taking as input the j^{th} feature map of the layer $l-1$, and generating as output the i^{th} feature map of the layer l.

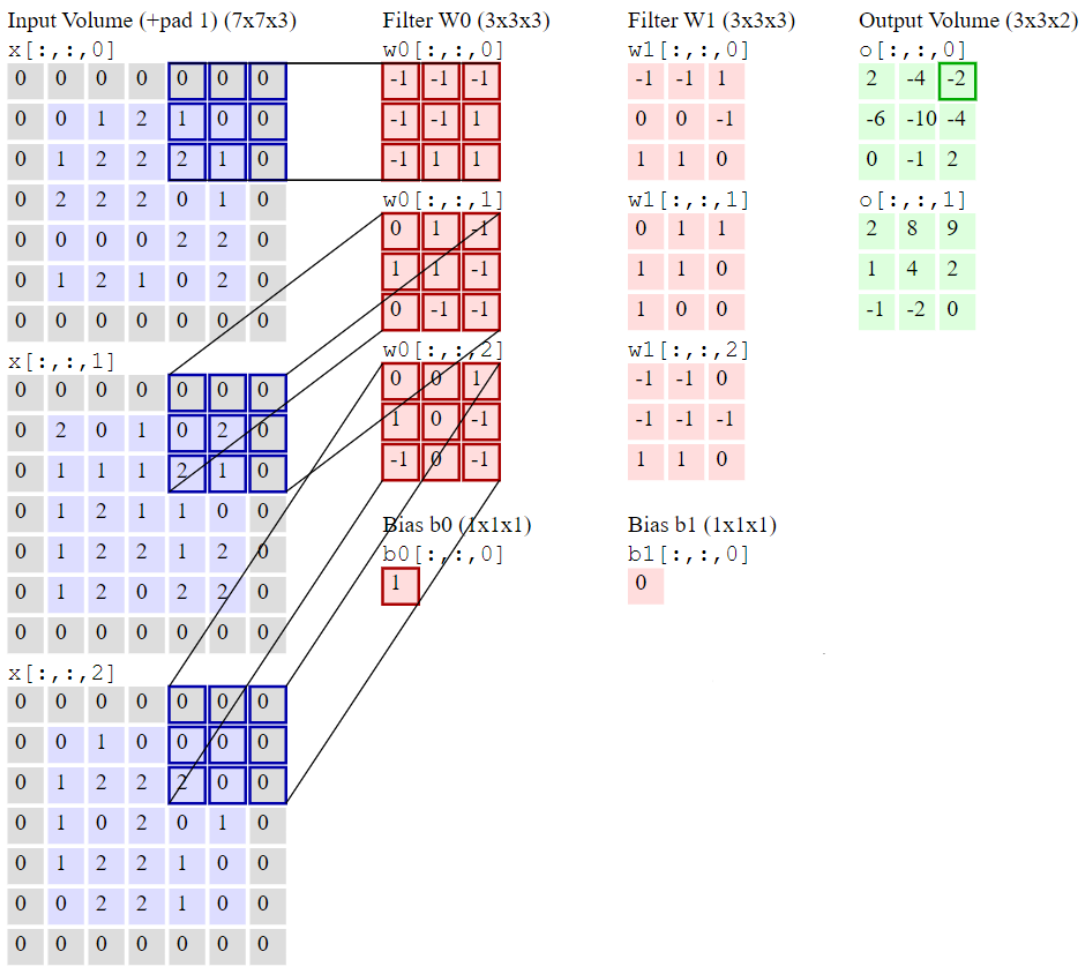


Figure 2.2: Visualizing the convolution operation. Element-wise multiplication of the input subsection and filter product which is summed to generate the target value at a specific position in the output. This figure was adapted from the course CS231n by Stanford University [10].

Each subsection of the image that goes through convolution is multiplied by the corresponding filter weight, and then the values are added to generate the value of one pixel of the feature map. Due to the fact that the filter size is predefined, and that the image can be of any size to perform convolution, CNNs are able to be scaled up or scaled down depending on the type of application [9]. This level of versatility is the reason for the CNNs success.

2.1.2. Activation Layer

The activation layer consists of a function that maps the output of the convolution onto a non-linear domain. Introducing these non-linearities is essential, since without the activation layers, a model could only learn linear decision boundaries, leading to poor performance [11]. The activation layer decides whether a neuron can fire or not, depending on if the neuron's value is above a threshold, known as the bias. Figure 2.3 shows what the different activation functions look like. Some of the common functions used in activations layers can be seen below.

- Sigmoid:

$$\sigma(z) = \frac{1}{1 + \exp^{\text{-z}}} \tag{2.2}$$

- Tanh:

$$\sigma(z) = \frac{\exp^{\text{z}} - \exp^{\text{-z}}}{\exp^{\text{z}} + \exp^{\text{-z}}} \tag{2.3}$$

- ReLU:

$$ReLU(z) = \begin{cases} z, & \text{z>0} \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

- Leaky ReLU:

$$LeakyReLU(z) = \begin{cases} z, & \text{z>0} \\ az, & \text{otherwise} \end{cases} \tag{2.5}$$

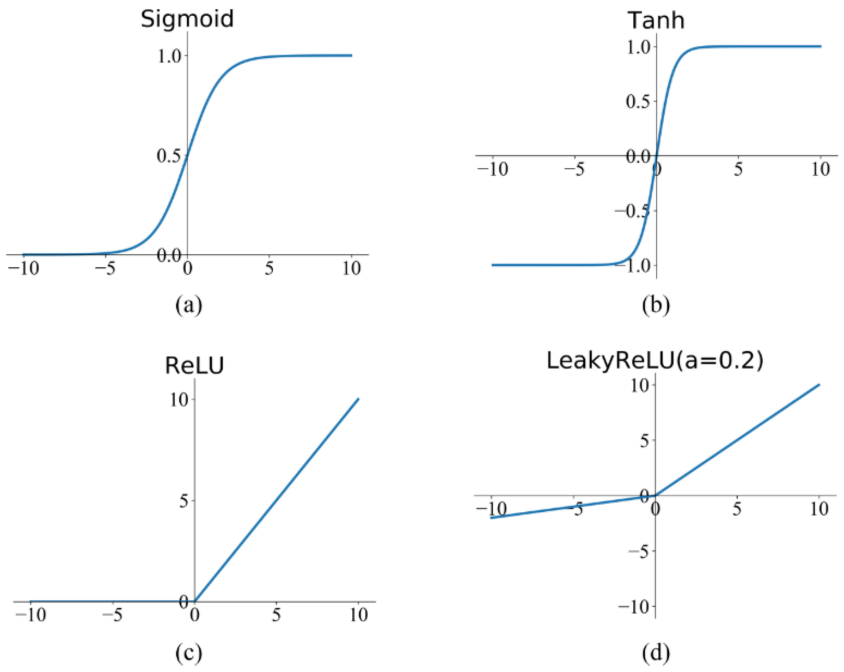


Figure 2.3: Common activation functions used to add non-linearities into deep learning models. The figures have been adapted from [12].

2.1.3. Pooling Layer

A pooling layer is typically used to reduce the size of the feature maps in order to make the CNN more computationally efficient. Pooling has a similar effect as sampling the image with a specified step size. This results in the image shrinking significantly but still maintaining most of the information [9]. Figure 2.4 shows the pooling layer in action. Similar to convolution, the pooling kernels convolve over the image and sample data depending on the type of pooling, shown below:

- Max Pooling: The largest pixel in the subsection of the image is chosen to be in the feature map of the down scaled image.
- Average Pooling: Average of all the pixels in the subsection of the image is chosen to be in the feature map of the down scaled image.
- Min Pooling: The smallest pixel in the subsection of the image is chosen to be in the feature map of the down scaled image.

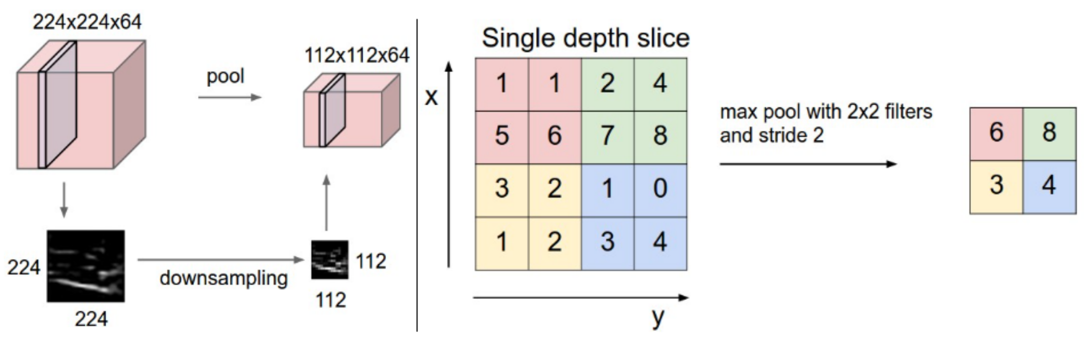


Figure 2.4: The figure shows how pooling is performed. When the stride of the filters is 2, the pooling layer has the same effect as downscaling the input by 2. This figure was adapted from the course CS231n by Stanford University [10].

2.1.4. Regularization

Regularization is necessary in order to prevent a model from performing too well on the training dataset and poorly on the test dataset. This is usually an indicator for overfitting, where the model has been over-tuned to the training data, and is extremely sensitive against any other data, causing poor results [13]. Regularization in the paper was performed by using Data Augmentation, which involves performing transformations on the data in order to artificially inflate the dataset. Doing so generates more samples, and makes it less likely for the model to overfit.

Deeper models are more likely to overfit, which is why they require immense amounts of data to be able to perform optimally. This is due to the fact that with each layer, the features get more and more complex, and if the amount of data is insufficient, the extracted features maps specifically to the provided data instead of generalizing [13]. Figure 2.5 shows the difference between a model that has overfit, fit correctly, and underfit.

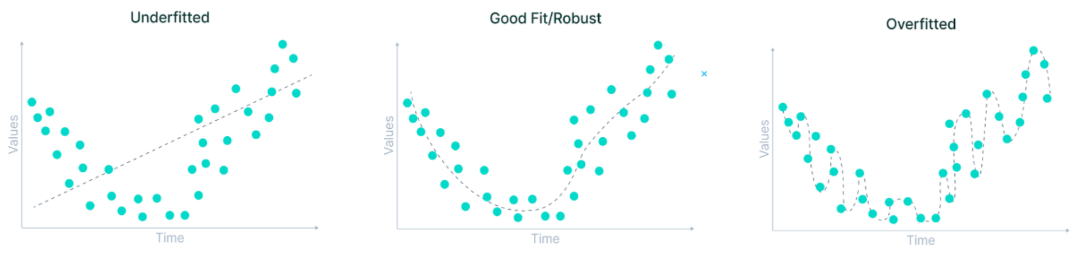


Figure 2.5: Increasing the complexity of the model increases the likelihood of overfitting. This figure was adapted from [14].

2.1.5. Fully Connected Layer

The role of a fully connected layer is to connect every neuron in the previous layer to every neuron in the next layer. Fully connected layers are used at the end of the CNN in order to convert the filters into class confidence weights, with which the predictions can be made [9]. The final layer of a CNN is typically a fully connected layer with the same number of neurons as the number of classes. The activation of each of the neurons tells us how confident the model is in the input image being from that class. Figure 2.6 shows a model with all the layers we have discussed so far.

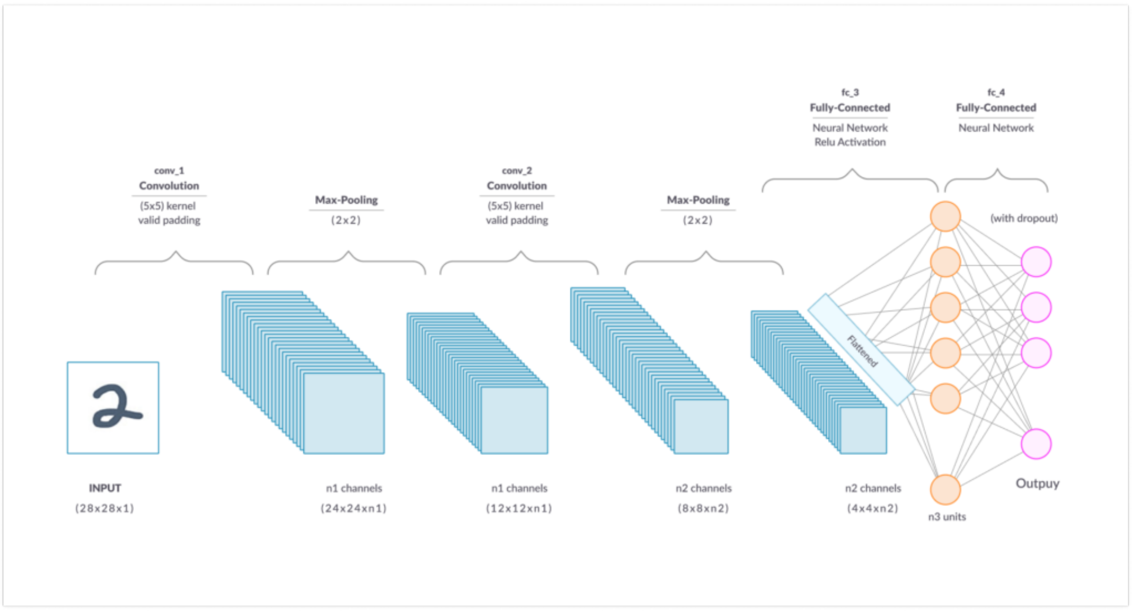


Figure 2.6: Increasing the complexity of the model increases the likelihood of overfitting. This figure was adapted from [15].

2.2. Resnet

Resnet is a family of CNNs that were introduced in 2015 [8] and have since had a significant contribution to deep learning research. They consist of residual connections, which eliminate the problem of vanishing gradients [16]. This problem occurred in deep networks as the deeper the network is, the more likely it is that the gradients will tend to zero due to the increased number of derivatives that need to be calculated.

2.2.1. The problems with deep CNNs

The paper introducing resnet [8] mentions that deep networks faced a problem with *degradation*, where as the layers increase, the training accuracy of the model starts to drop. This is exactly the opposite as was expected since increasing the layers tends to increase overfitting, which results in unexpectedly high training accuracy. This problem is a result of vanishing gradients. Resnet solves this issues by having shortcut connections, which is an identity mapping that allows the gradients to propagate deeper into the network. This resulted in Resnets improving drastically from depth in comparison to other networks, giving rise to networks like ResNet-18, ResNet-34, ResNet-152, where the number represents the number of layers. A standard ResNet-18 has been shown in Figure 2.7.

2.2.2. Simplicity of Resnet

Resnet had a top-5 error rate of 3.57% on Imagenet, which was a significant improvement given the fact that Resnet was less complex than other networks [8]. The shortcut connections allowed for models that could be far simpler. In our paper, we have used Resnet for the same reason. Recurrent models can be made to be simpler than current CNNs due to increased biological realism by adding additional connections between layers. Due to this, Resnet was the most compatible model with the changes we intended to make.

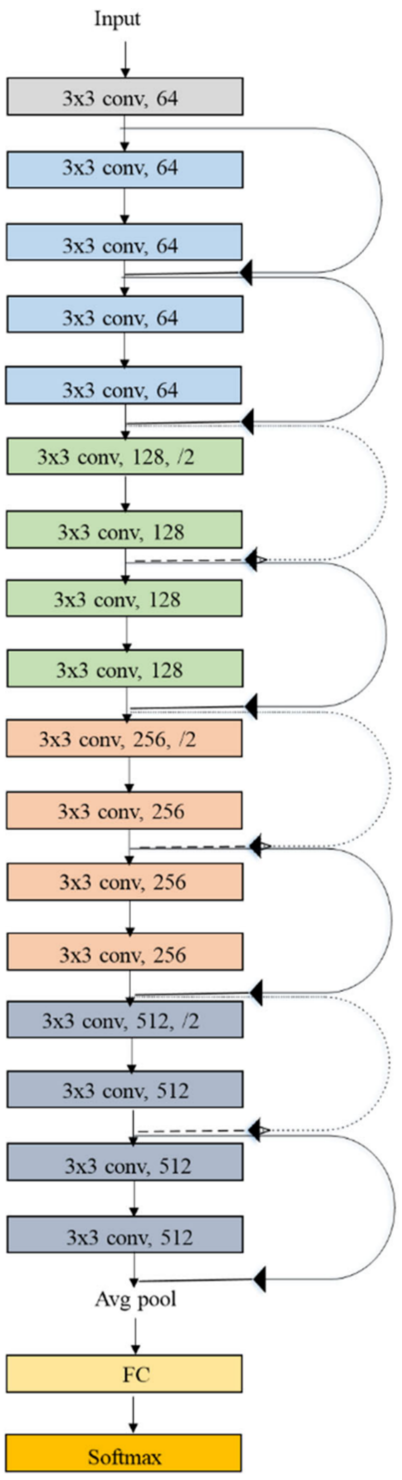


Figure 2.7: The figure shows the architecture of a standard ResNet-18 model. The figure has been taken from the paper by Ramzan et al [17].

3 Image Classification

Image classification is a part of Supervised Learning in Machine Learning [18]. This means that the data we train the model on has two parts: the input, and the prediction. The input in our case, is an image from our dataset, and the prediction is the class of the image. The CNN learns these classifications by learning from a extensive amount of examples, to form features that are successfully able to differentiate between classes [5]. Figure 3.1 shows us a visual for how the features extracted by the CNN can be seen as a weighted average of the examples seen by the CNN, weighted by their relative importance towards improving the accuracy of the model.

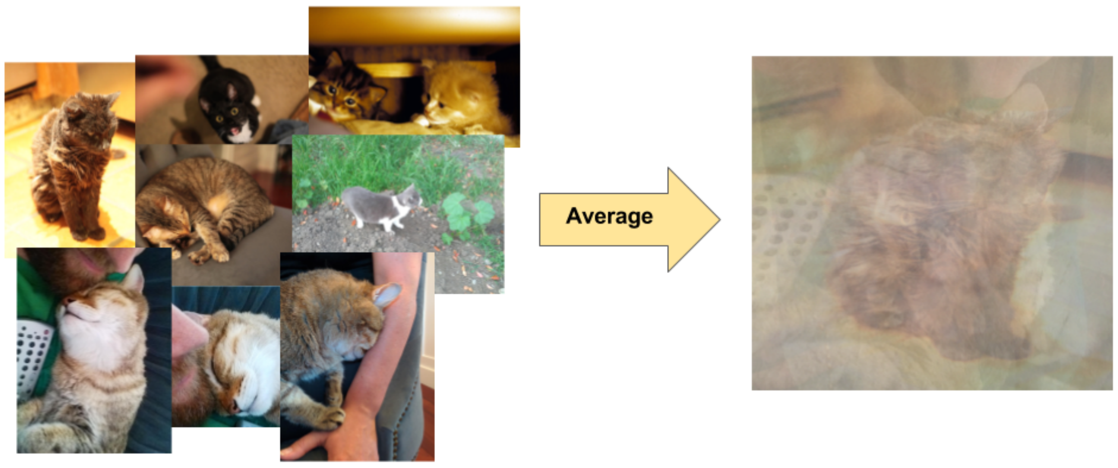


Figure 3.1: The resulting image may look meaningless in terms of representing a cat, the the image is a result of capturing the most important features necessary to identify whether something is a cat or not. These features are all combined into the feature map we see. This figure was adapted from [19].

3.1. Softmax Classifier

The output of the final fully connected layer is a set of values that correspond to the model's confidence in the class prediction. These values are independent of each other and the higher the value is, the more likely the prediction is to be correct [20]. Using a softmax classifier takes all the values of the fully connected layer, and transforms them to sum to 1 [20]. This results in each value representing the probability with which the model thinks the prediction is correct. Doing so makes it possible to use a loss function to calculate how good the predictions were. This loss value is crucial in ensuring that the model reaches optimal performance [21]. Softmax can be applied to a vector of values, which results in the values being transformed to be between 0 and 1, such that all the values sum to 1. It can be performed using the formula below:

$$\sigma(X)_{\mathrm{i}} = \frac{\exp(X_{\mathrm{i}})}{\sum_{j=1}^{K} \exp(X_{\mathrm{j}}} \tag{3.1}$$

Where $\exp(X_{\mathrm{i}})$ is the value at the i^{th} position of the vector
Figure 3.2 shows the result of converting a set of values to a set of class predictions.

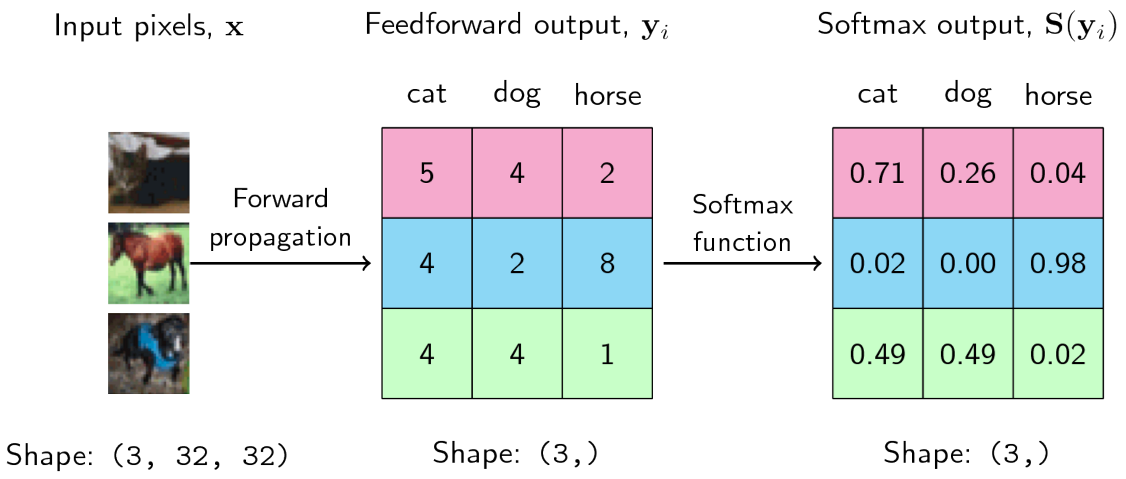


Figure 3.2: The values of the feedforward layers are transformed into class probabilities using the softmax function. This figure was adapted from [22].

3.2. Loss Function

A loss function allows us to see whether a model is performing better or worse over time. In our paper, we used the loss function called categorical cross entropy. This can be calculated as seen below:

$$Loss = -\sum_{i=1}^{N} y_{\mathrm{i}} * log(\hat{y}_{\mathrm{i}} \tag{3.2}$$

Where N is the sample size of the data, y_{i} is the actual prediction, and $\hat{y}_{\mathrm{i}}$ is the model prediction.

When the prediction starts to match the actual class, the absolute value of the Loss will start increasing. The Loss value contains a negative sign as we want it to be very small when the model is performing well. This loss value is then used to calculate the filter weights using backpropagation [23].

3.3. Backpropagation

Backpropagation is the method with which the gradients are calculated for all the weights using the loss value, and these gradients propagate backwards into the model as we use the gradients of the current layer to calculate those of the previous layer [23]. To optimize the loss function, we use gradient descent, which involves calculating the gradient at the current time step, and using that to tune the weights slightly so the model has a slightly lower loss value in the next time step. Over many steps, the model is able to converge, and a minimum value is found for the loss function [24].

3.3.1. Gradient Descent

Figure 3.3 an example of what gradient descent looks like over multiple time steps. We see that for each time step, the gradient will always lead down the slope, with each step getting smaller and smaller as we get closer to the minimum. It is important to tune this step size, as a step size that is too small will lead to extremely long computation time to optimize the model [24]. Additionally, as the step size gets smaller, the gradient gets smaller as well until is becomes too small to be meaningful. This problem is known as vanishing gradient [16] that ResNet eliminates by using residual connections, explained earlier. A step size that is too big will cause the model to jump around the loss function, missing the minimum and causing the weights to become too large. This problem is known as exploding gradient [25], and is a sign that the model needs to reduce the step size.

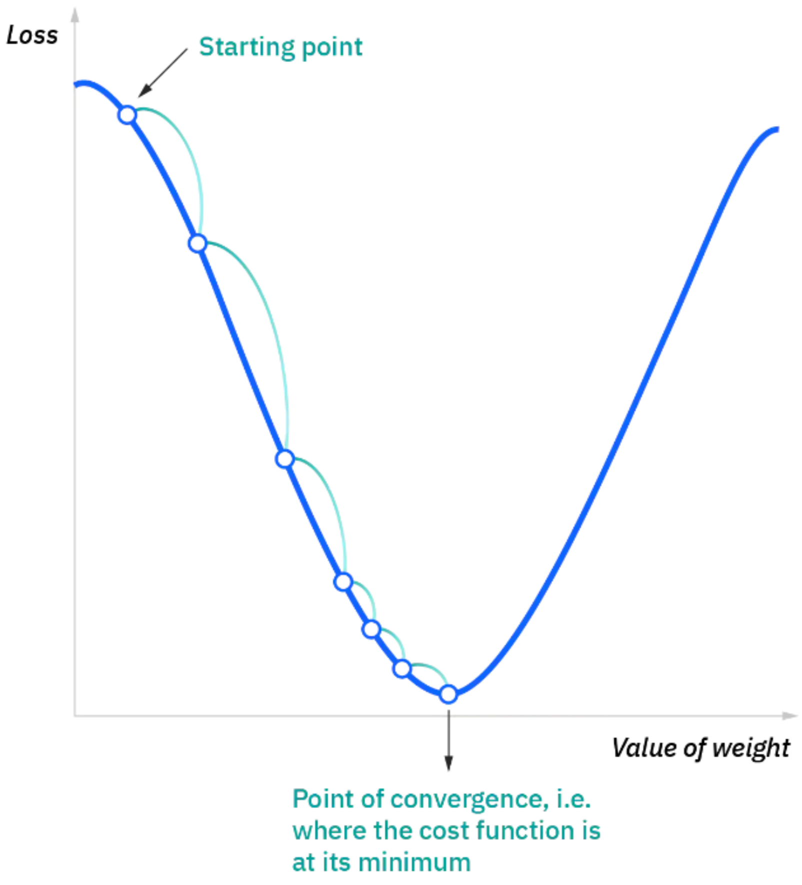

Loss
Starting point
Value of weight
Point of convergence, i.e. where the cost function is at its minimum

Figure 3.3: We see that with each step, the model gets closer to an optimum. This figure was adapted from [26].

3.4. Classification Overview

The classification process requires the following steps, discussed in the previous chapter:

- Model receives an input
- Convolution Layer applies various filters to the data and generates feature maps
- Activation function introduces non-linearity into the feature maps
- Pooling layer downsamples the feature maps in order to improve computational efficiency and reduce the memory usage
- Fully connected Layer combines the features and generates output depending on the class the image belongs to
- Softmax classifer generates the prediction along with the confidence for each prediction
- All the steps above constitute one forward pass. Following the forward pass, a backward pass takes place where backpropagation is used to update the filter weights until an optimal value is reached.

4
Biological Realism: Comparing models with brains

Given that Neural Networks were inspired by biological brains as well [27], it is only natural to expect bio-inspired models all throughout Deep Learning architecture. Having bio-inspired models allows us to make analogies between the brain and the model in terms of their behavioural, structural, or internally observable similarities, benefiting both the field of computer science, and neuroscience [28], [29].

The primary visual system can be divided into 4 major regions, V1, V2, V4, and IT [30]. These regions are similar to having convolution blocks in a CNN, where the features extracted get increasingly complex after each block. These complex features are then used by IT for inference [1]. Figure 4.1 shows what the hierarchy of the visual system looks like. We can see that the models have a feed-forward hierarchy where V1 gives information to V2, which sends the information to V4, which finally passes the information to IT. But we also see recurrent connections, causing time to be a crucial factor for robust vision [31].

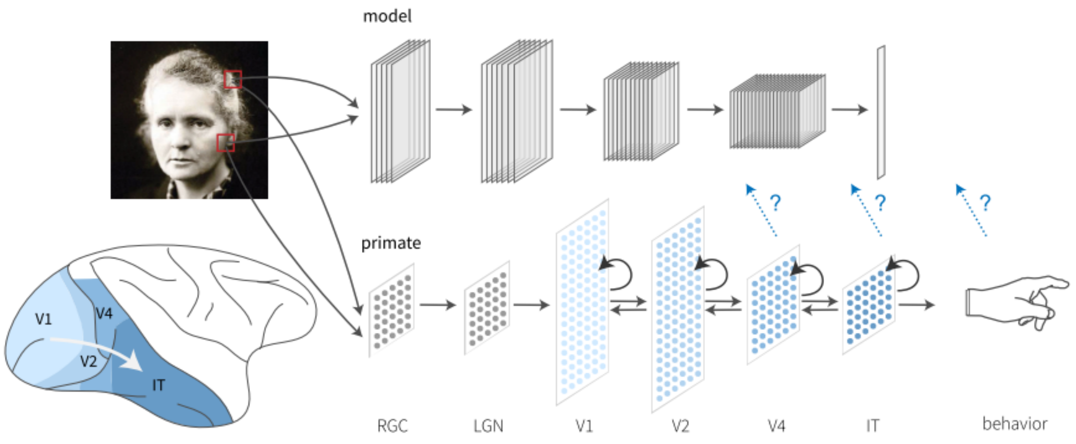


Figure 4.1: The figure shows how structural analogies between a CNN and the primary visual system can be made. The figure was taken from the paper by Schrimpf et al [32].

4.1. BrainScore

One way to compare models to the brain would be to use a benchmark known as BrainScore [32]. This benchmark consists of neural metrics, which compare how similar the neuronal activations of the computational model are to prerecorded neuronal activations of primate subjects. Another metric used by BrainScore is a behavioural metric, which compares how similar the predictions of the model are to the predictions made by human subjects. This allows us to assess whether a computational model is susceptible to the same mistakes, or if the model also experiences some variation of an optical illusion like humans do. The goal of BrainScore is to produce models that aren't necessarily better in terms of performance, but are ever-closer approximations of the primate visual system. However, performance and BrainScore are positively correlated, so a model that does well on ImageNet is likely to have a good BrainScore, as seen in Figure 4.2.

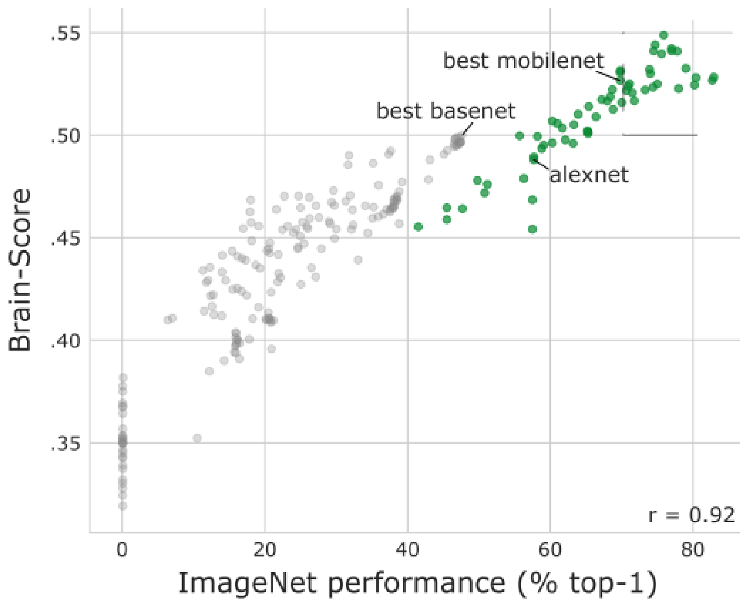


Figure 4.2: The figure shows how BrainScore and the performance on Imagenet are correlated. The figure was taken from the paper by Schrimpf et al [32].

4.1.1. Effect of recurrence on BrainScore

We used the same recurrent model used in the experiments in the paper to calculate its brainscore. This would tell us whether recurrence has any effect on whether a model is more brain-like. Table 4.1 shows us the results of this experiment. We see from the table that adding recurrence has a +29%, which is to be expected as the intention behind adding the recurrent connections was to create a more brain-like model by replicating parts of its structure. The coarse to fine model performed even better, which indicates that the additional blurring provides some behavioral similarity to the brain. It is possible that the blurring induced a stronger shape bias in the model, which resulted it in being more brain-like [33].

Model Name	BrainScore	Change in Score (wrt baseline)
Resnet (baseline)	0.185	+ 0%
Recurrent Resnet	0.239	+ 29%
Recurrent Resnet CtF	0.251	+ 36%

Table 4.1: The table shows the change in brainscore once recurrent connections were added to the baseline model.

4.1.2. Problems with BrainScore

As stated earlier, Brainscore is correlated to ImageNet accuracy [32]. This means that to get accurate Brainscores, the models need to been trained on ImageNet, which has a high computational cost. In our experiments, our models were trained on cifar10 [34], which does not provide enough diversity in comparison to ImageNet in classes to have a high Brainscore. However, since all the models had undergone the same training, we can still compare the results relative to one another. Doing so has it's own additional problems though. The Brainscore tends to vary considerably for smaller models trained on alternate datasets. Table 4.2 shows the variance in Brainscores of the same resnet model trained on different seeds. Seeing how much the scores can vary, going as high as a 20% increase, it becomes difficult to differentiate if the results observed in other experiments are due to changes in the model, or due to noise. Thus, we can only make conclusions for models that show drastically different scores, like in Table 4.1.

Model Seed	BrainScore	Change in Score (wrt baseline)
Seed 05 (baseline)	0.185	+ 0%
Seed 10	0.214	+ 16%
Seed 15	0.211	+ 14%
Seed 20	0.222	+ 20%
Seed 25	0.202	+ 9%
Seed 30	0.197	+ 6%

Table 4.2: The table shows the change in brainscore as we retrain the same model on a different seed.

4.2. Recurrent CNNs

One shortcoming of CNNs for modeling the brain is that CNNs are feedforward models while the brain has recurrent connections [29], [35]. Therefore, a bio-inspired CNN that contains these recurrent connections may be able to learn similar mechanisms as the visual system [36], [37]. However, traditional recurrent neural networks do not display recurrence in the same manner that is found in the recurrent connections of the brain [39]. The information passes through the brain's hierarchy sequentially, in a stepwise manner. This would require the recurrent connections to be modified to be biologically consistent. We can see the difference by referring to Figures 4.3 and 4.4. Figure 4.4 performs the biologically consistent recurrent computations in a way inspired by a model named CORnet, introduced in the paper by Kubilius et al [39].

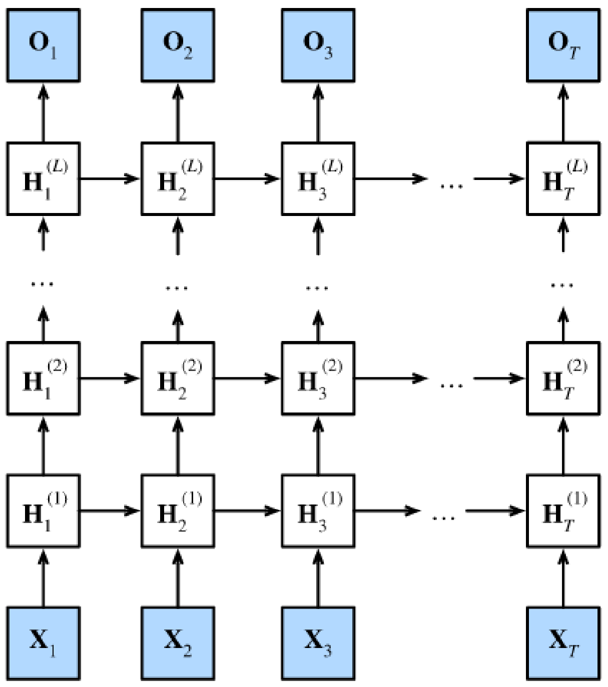


Figure 4.3: The figure shows a common recurrent neural network, when unrolled in time. We can see that the input directly influences the output in the same time step. This figure was adapted from [38].

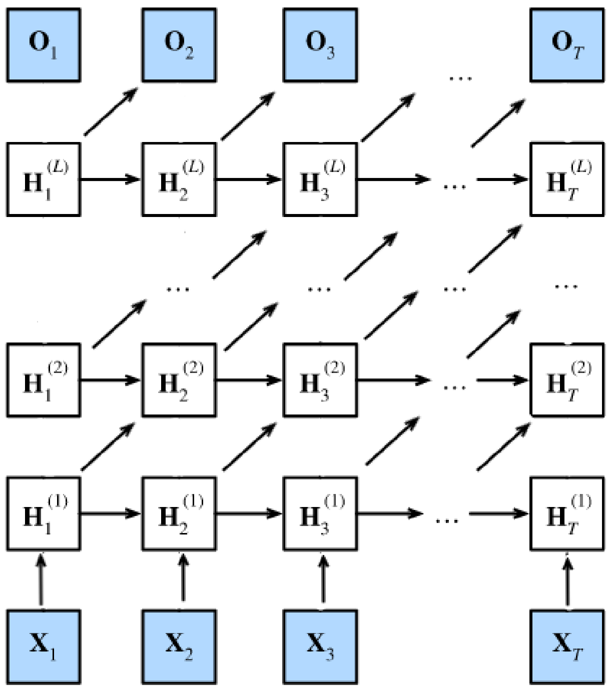


Figure 4.4: This figure shows a recurrent network unrolled in a biologically realistic manner. The computations are stepwise, and the influence of the input takes a few time steps to reach the output.This figure was adapted from [38].

4.2.1. CORnet

CORnet [39] is a family of biologically-realistic recurrent models. Figure 4.5 shows a recurrent unit used in one of the CORnet models, which performs computations in the order shown in Figure 4.4. CORnet was shown to have the same BrainScore as AlexNet [32] and was an important contribution to the field of computational neuroscience. The recurrent mechanism of CORnet allowed us to modify a ResNet to show how recurrence affects robustness. The CORnet models generally consist of four of the recurrent units shown in Figure 4.5, where each unit represents one subsection of the hierarchy of the primary visual system. The units are anologous to the visual system's V1, V2, V4, and IT regions in order to learn similar behaviour as the brain as they are optimized to improve their brainscore.

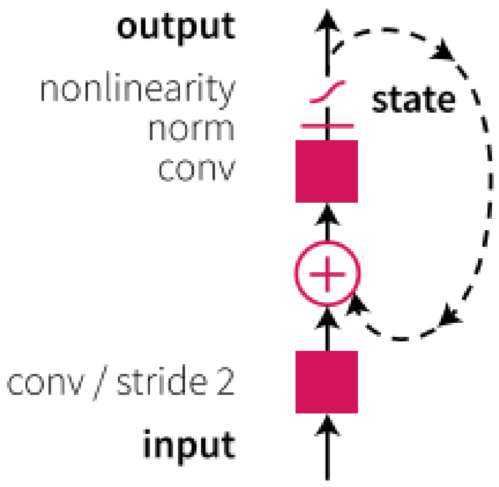


Figure 4.5: The figure has been taken from the paper by Kubilius et al [39]. It shows a recurrent unit used in CORnet.

5 Additional Experiments: Random Erasing

5.1. Motivation

Random Erasing was an additional adversarial attack that was applied on the models in the paper. This attack worked by removing rectangular a section from the image, leaving the image with a black patch. The proportion of image that could be erased was a hyperparameter. During training time, only 5% of the image was erased, but the proportion went up to 40% during test time. Figure 5.1 shows an example of the different proportions that the model had to predict during test time.

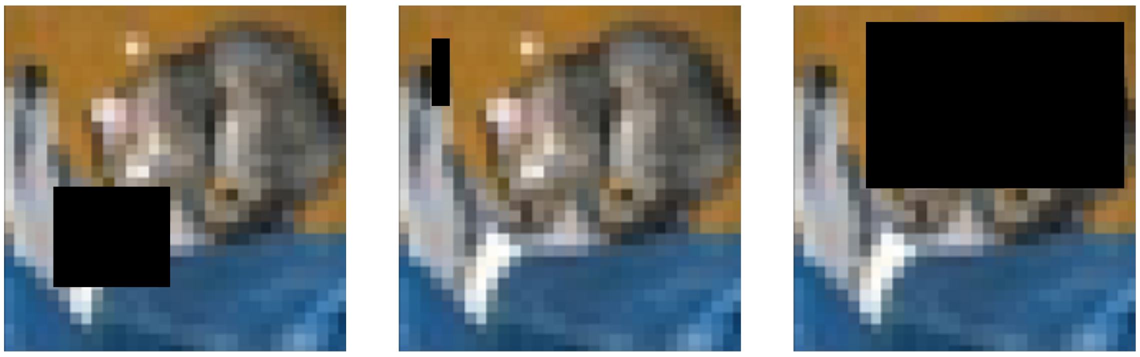


Figure 5.1: Larger patches were erased during test time to assess whether the models were able to generalize to the attack with minor examples or if the models overfit on the specific intensity of the attack.

This attack was chosen as recurrence is said to help early layers fill in missing information by receiving contextual information from deeper layers [35], [40], which would mean that the recurrent models would have an advantage in situations where data might be missing. Figure 5.2 shows the result of the experiment.

5.2. Results

We can see from Figure 5.2 that when the models are not adversarially trained, the recurrent models perform worse than the standard resnet model. This indicates that the missing information did not receive enough contextual information to make up for the erased patches. After adversarial training, all three models had somewhat similar responses, with the recurrent model slightly outperforming the standard resnet model. This would show that despite being worse without adversarial training, the recurrent model generalizes better, and is able to maintain its accuracy with stronger attacks.

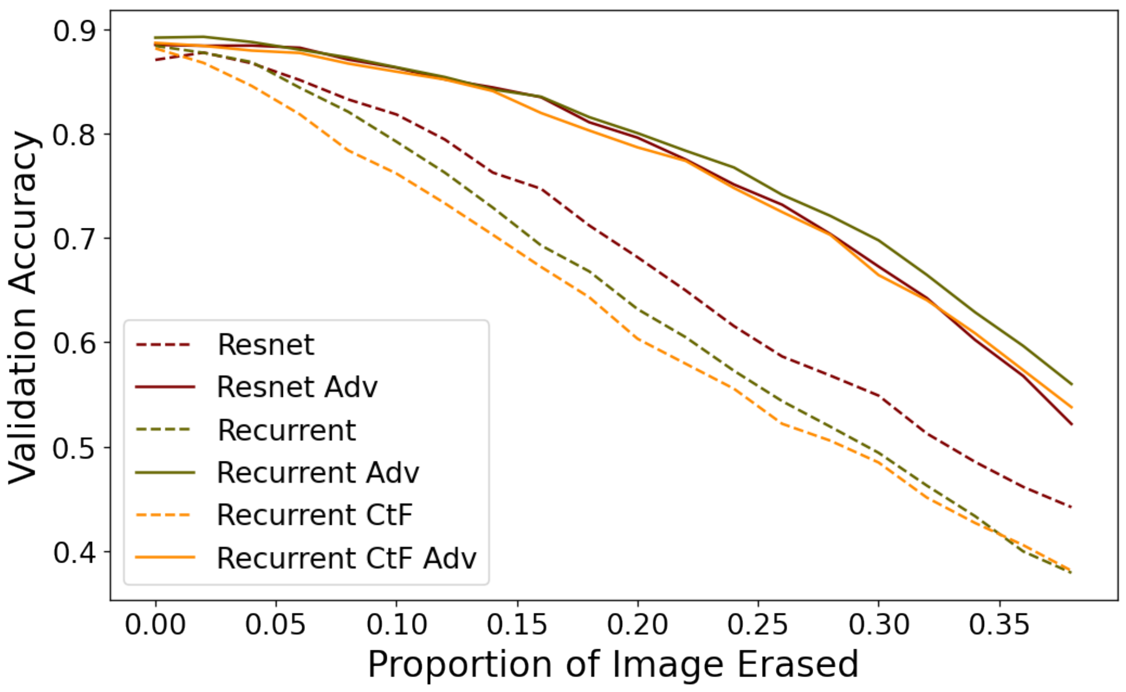


Figure 5.2: As the attack gets more intense, we see the validation accuracy start to drop for all models. Some models drop faster than others depending on their resistance to that particular type of adversarial attack.

5.3. Conclusion

Considering our hypothesis that the model is able to increase robustness through spatio-temporal averaging, the results are to be expected. Since the attack takes away considerable patches of the image, the recurrent connections are unable to average out such a locally concentrated attack of a high magnitude in comparison to the minor perturbations caused by Gaussian noise. This would mean that recurrence does not provide any substantial advantage against attacks of this particular type and thus the attack affects both the resnet and the recurrent model relatively similarly.

6 Additional Experiments: Rotation

6.1. Motivation

We decided to use rotations as our hypothesis was testing whether recurrent models gain robustness through spatio-temporal averaging. Since averaging over time steps will not get rid of any effects of rotations, we would expect to see no improvement, or some decrease in the performance of the recurrent models. The adversarially trained models included samples that were rotated anywhere between 0 degrees to 5 degrees, and during test time, these models were attacked with rotations from 0 degrees to 45 degrees. This was done to test the model's ability to generalize to an attack. Figure 6.1 shows the transform used on a sample image.

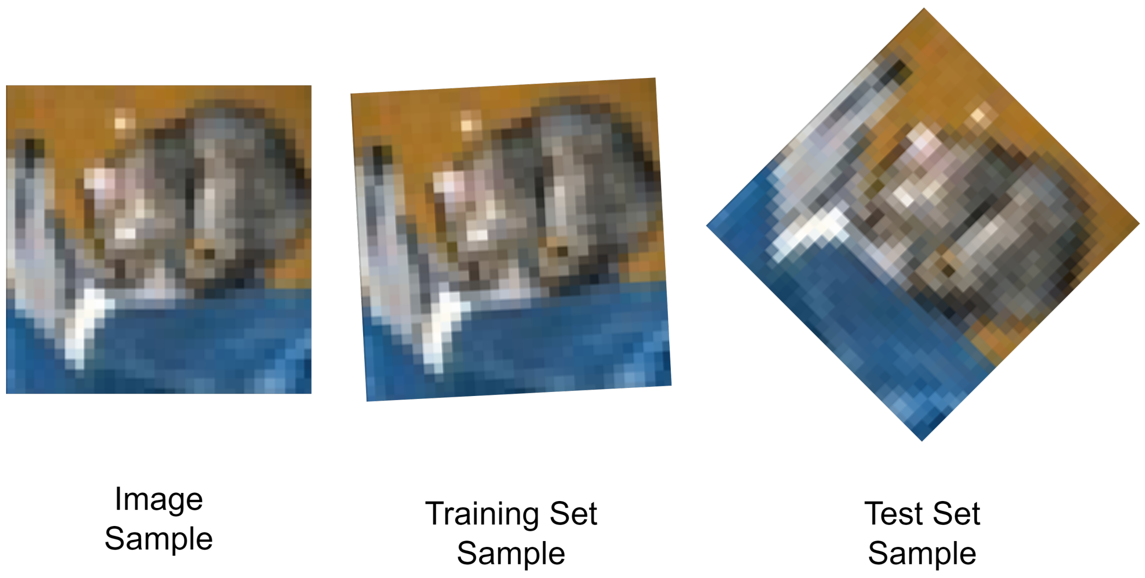


Figure 6.1: The images show the maximum rotation allowed for the adversarial training (middle) and the adversarial attack (right) on the target image (left).

6.2. Results

We can see from Figure 6.2 that the baseline resnet is the most robust when the models are not adversarially trained, followed closely by the coarse to fine recurrent model. A reason for why the coarse to fine model is able to keep up with the resnet model for lower values of rotation may be because rotations affect the image more near the edges. This would mean that for lower rotation values, the blurring in the model would cause the central regions to still be recognizable. After the models are adversarially trained and attacked again, we see that the models all perform relatively similarly, until finally resnet outperforms the other models for high rotations.

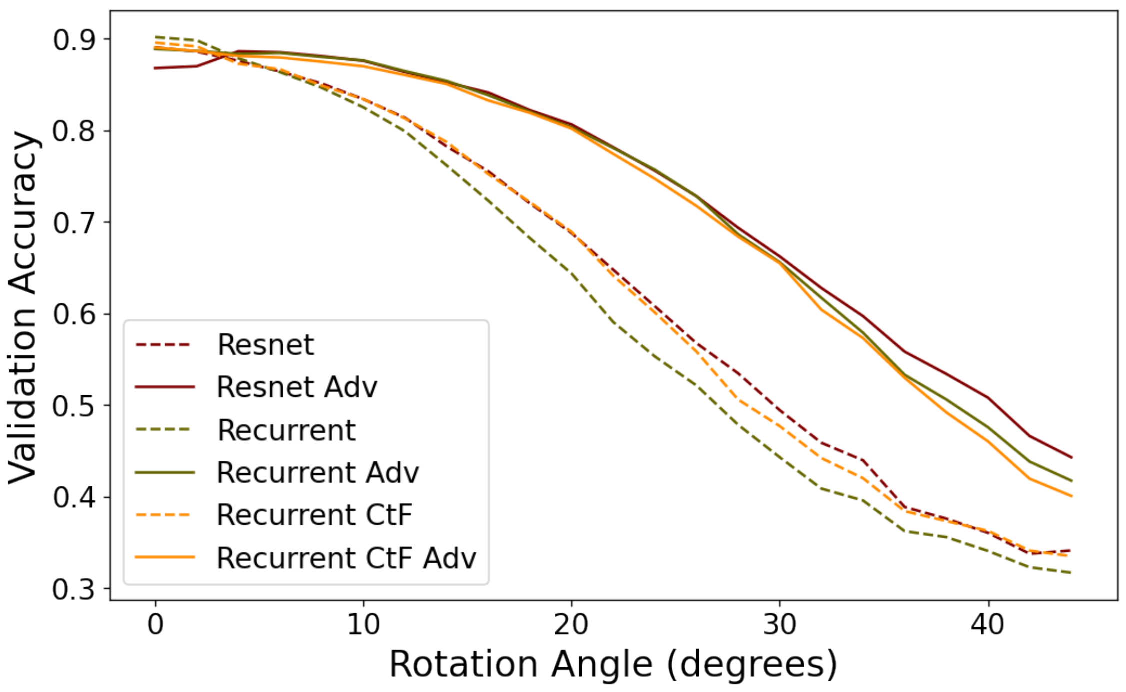


Figure 6.2: We see that the adversarially trained models perform identically until the recurrent models drop for high rotations..

6.3. Conclusion

As stated before, this result is expected as spatio-temporal averaging cannot aid robustness for transforms like robustness, so we can not expect any drastic improvements. For high rotations, the contextual information that the layer receives could correspond to different areas in the actual image, resulting in a drop in accuracy. However, the results are not too significant and need to be repeated or redone differently to produce results with more definitive trends.

7 Additional Experiments: Perlin Noise

7.1. Motivation

Perlin noise was used as it is a correlated noise and cannot be averaged over to reduce the effect of the noise [41]. This is only a property of uncorrelated noise with a zero mean. Due to this, we would expect that the recurrent model would not be better than the resnet model. We did not perform adversarial training for this attack as we wanted to see the effects on models that were not adversarially trained first. Figure 7.1 shows what an image corrupted by perlin noise looks like. During training time, the weight of the perlin noise was limited to 5% of the input image intensity, but this weight went up to 50% of the image intensity during test time.

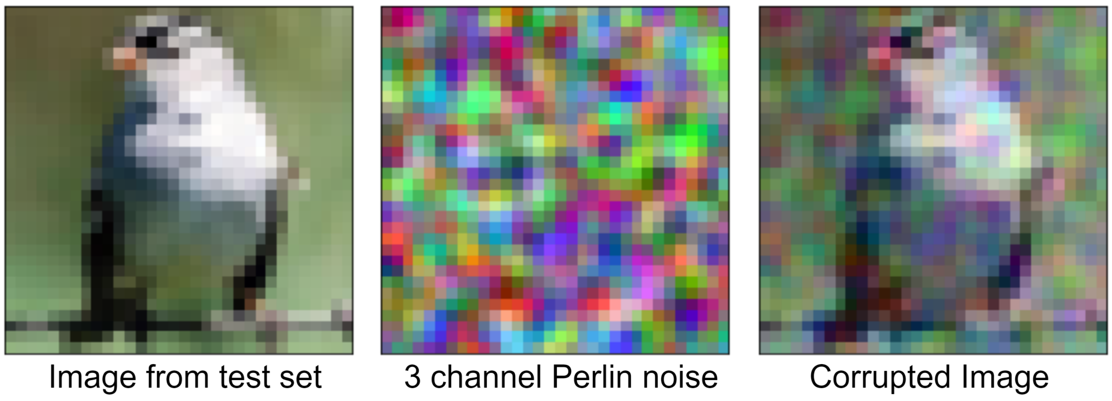


Figure 7.1: Example of bird affected by perlin noise. The noise is smoother than Gaussian noise as it is correlated, so the transitions will be smoother. The noise in the image is exaggerated for visualization purposes.

7.2. Results

We can tell by looking at Figure 7.2 that the models have performed roughly equally. The recurrent model slightly outperforms the other models, but that could also be due to the fact that the model also had a slightly higher accuracy when there is no noise. The models have performed similarly as the difference between the accuracy remains almost equal as well, indicating that no model had any particular advantages during the adversarial attack. Additionally, the computational expense of generating perlin noise in 4 dimensions (Image height, Image width, No. of channels, No. of samples) made this experiment extremely time consuming to scale up. For further experimentation with perlin noise, more efficient algorithms would be necessary.

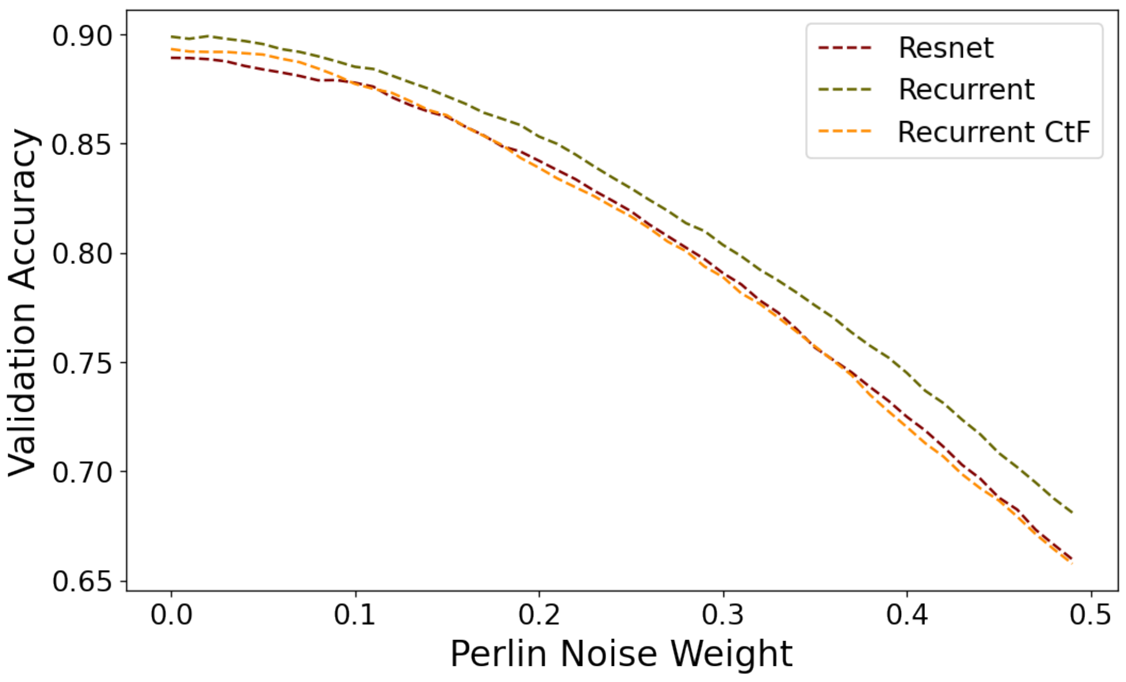


Figure 7.2: We can see as the attack intensifies, all models drop in accuracy roughly at the same rate.

7.3. Conclusion

The Figure 7.2 shows results that are not significant enough to support our hypothesis, though it shows scope for future researchers to analyze the effects of different types of correlated noise. Further experiments could include trying out perlin noise of different frequency so the resulting noise can be made to be rougher or smoother, emulating the grainy nature of gaussian noise, or the smooth transitions of the correlated noise used in the paper. This would further allow one to study whether the frequency of the noise is a significant factor that influences robustness as well.

List of Figures

List of Tables

Bibliography

[1] D. L. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo, “Performance-optimized hierarchical models predict neural responses in higher visual cortex,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, no. 23, pp. 8619–8624, 2014, ISSN: 10916490. DOI: 10.1073/pnas.1403112111.

[2] J. H. Jacobsen, J. V. Gemert, Z. Lou, and A. W. Smeulders, “Structured Receptive Fields in CNNs,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 2610–2619, 2016, ISSN: 10636919. DOI: 10.1109/CVPR.2016.286.

[3] S. Wang, T. Liu, and L. Tan, “Automatically learning semantic features for defect prediction,” *Proceedings - International Conference on Software Engineering*, vol. 14-22-May-, pp. 297–308, 2016, ISSN: 02705257. DOI: 10.1145/2884781.2884804.

[4] A. R. Pathak, M. Pandey, and S. Rautaray, “Application of Deep Learning for Object Detection,” *Procedia Computer Science*, vol. 132, no. Iccids, pp. 1706–1717, 2018, ISSN: 18770509. DOI: 10.1016/j.procs.2018.05.144. [Online]. Available: https://doi.org/10.1016/j.procs.2018.05.144.

[5] K. Patel, K. Rambach, T. Visentin, D. Rusev, M. Pfeiffer, and B. Yang, “Deep learning-based object classification on automotive radar spectra,” *2019 IEEE Radar Conference, RadarConf 2019*, 2019. DOI: 10.1109/RADAR.2019.8835775.

[6] R. Hou, C. Chen, and M. Shah, “Tube Convolutional Neural Network (T-CNN) for Action Detection in Videos,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 5823–5832, 2017, ISSN: 15505499. DOI: 10.1109/ICCV.2017.620.

[7] T. J. Sejnowski, “The unreasonable effectiveness of deep learning in artificial intelligence,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 117, no. 48, pp. 30 033–30 038, 2020, ISSN: 10916490. DOI: 10.1073/pnas.1907373117.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016, ISSN: 10636919. DOI: 10.1109/CVPR.2016.90.

[9] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, vol. 2018-Janua, pp. 1–6, 2018. DOI: 10.1109/ICEngTechnol.2017.8308186.

[10] F.-F. Li, J. Wu, and R. Gao, *CS231n Convolutional Neural Networks for Visual Recognition*, Jan. 2022. [Online]. Available: https://cs231n.github.io/convolutional-networks/.

[11] Y. Wang, Y. Li, Y. Song, and X. Rong, “The influence of the activation function in a convolution neural network model of facial expression recognition,” *Applied Sciences (Switzerland)*, vol. 10, no. 5, 2020, ISSN: 20763417. DOI: 10.3390/app10051897.

[12] J. Feng, X. He, Q. Teng, C. Ren, H. Chen, and Y. Li, “Reconstruction of porous media from extremely limited information using conditional generative adversarial networks,” *Physical Review E*, vol. 100, no. 3, p. 33 308, 2019, ISSN: 24700053. DOI: 10.1103/PhysRevE.100.033308. [Online]. Available: https://doi.org/10.1103/PhysRevE.100.033308.

[13] J. Kukačka, V. Golkov, and D. Cremers, “Regularization for Deep Learning: A Taxonomy,” pp. 1–23, 2017. [Online]. Available: http://arxiv.org/abs/1710.10686.

[14] P. Baheti, *Overfitting vs Underfitting in Machine Learning [Differences]*, Jul. 2022. [Online]. Available: https://www.v7labs.com/blog/overfitting-vs-underfitting.

[15] IndianTechWarrior, *Fully Connected Layers in Convolutional Neural Networks – IndianTechWarrior*, Apr. 2021. [Online]. Available: https://indiantechwarrior.com/fully-connected-layers-in-convolutional-neural-networks/.

[16] H. H. Tan and K. H. Lim, “Vanishing Gradient Mitigation with Deep Learning Neural Network Optimization,” *2019 7th International Conference on Smart Computing and Communications, ICSCC 2019*, pp. 0–3, 2019. DOI: `10.1109/ICSCC.2019.8843652`.

[17] F. Ramzan, M. U. G. Khan, A. Rehmat, *et al.*, “A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer’s Disease Stages Using Resting-State fMRI and Residual Neural Networks,” *Journal of Medical Systems*, vol. 44, no. 2, 2020, ISSN: 1573689X. DOI: `10.1007/s10916-019-1475-2`.

[18] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” *ACM International Conference Proceeding Series*, vol. 148, pp. 161–168, 2006. DOI: `10.1145/1143844.1143865`.

[19] Google, *ML Practicum: Image Classification | Machine Learning | Google Developers*, Jul. 2022. [Online]. Available: `https://developers.google.com/machine-learning/practica/image-classification`.

[20] B. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Cnn□□□□□,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2012.

[21] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, “On optimization methods for deep learning,” *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pp. 265–272, 2011.

[22] L. Miranda, *Understanding softmax and the negative log-likelihood*, Aug. 2017. [Online]. Available: `https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/`.

[23] G. Montavon, G. G. B. Orr, K.-R. Müller, *et al.*, *Neural Networks: Tricks of the Trade*. 2012, p. 432, ISBN: 3540653112. [Online]. Available: `http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Neural+Networks:+Tricks+of+the+Trade#3`.

[24] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7700 LECTU, pp. 437–478, 2012, ISSN: 16113349. DOI: `10.1007/978-3-642-35289-8{\_}26`.

[25] R. Y. Sun, “Optimization for Deep Learning: An Overview,” *Journal of the Operations Research Society of China*, vol. 8, no. 2, pp. 249–294, 2020, ISSN: 21946698. DOI: `10.1007/s40305-020-00309-6`.

[26] IBM, *What is Gradient Descent? | IBM*, Oct. 2020. [Online]. Available: `https://www.ibm.com/cloud/learn/gradient-descent`.

[27] G. W. Lindsay, “Convolutional neural networks as a model of the visual system: Past, present, and future,” *Journal of Cognitive Neuroscience*, vol. 33, no. 10, pp. 2017–2031, 2021, ISSN: 15308898. DOI: `10.1162/jocn{\_}a{\_}01544`.

[28] T. C. Kietzmann, M. R. C. Cognition, and B. S. Unit, “Deep Neural Networks in Computational Neuroscience Explaining Brain Information Processing Requires Complex , Task-,” *Oxford Research Encyclopaedia of Neuroscience*, no. January, pp. 1–29, 2021.

[29] R. Rajalingham, E. B. Issa, P. Bashivan, K. Kar, K. Schmidt, and J. J. DiCarlo, “Large-scale, high-resolution comparison of the core visual object recognition behavior of humans, monkeys, and state-of-the-art deep artificial neural networks,” *Journal of Neuroscience*, vol. 38, no. 33, pp. 7255–7269, 2018, ISSN: 15292401. DOI: `10.1523/JNEUROSCI.0388-18.2018`.

[30] W. Osberger, A. Maeder, and D. McLean, “A Computational Model of the Human Visual System for Image Quality Assessment,” *Dicta-97*, pp. 337–342, 1997.

[31] G. F. Elsayed, N. Papernot, S. Shankar, *et al.*, “Adversarial examples that fool both computer vision and time-limited humans,” *Advances in Neural Information Processing Systems*, vol. 2018-Decem, no. NeurIPS, pp. 3910–3920, 2018, ISSN: 10495258.

[32] M. Schrimpf, J. Kubilius, H. Hong, *et al.*, “Brain-Score: Which Artificial Neural Network for Object Recognition is most Brain-Like?” *bioRxiv*, pp. 1–9, 2018, ISSN: 2692-8205. DOI: `10.1101/407007`.

[33] S. Yoshihara, T. Fukiage, H. Information, N. Telegraph, and T. Corporation, "DO TRAINING WITH BLURRED IMAGES MAKE CONVOLUTIONAL NEURAL NETWORKS CLOSER TO HUMANS CONCERNING OBJECT RECOGNITION PERFORMANCE AND INTERNAL REPRESENTATIONS ?," 2022.

[34] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, pp. 1–9, 2010. [Online]. Available: `http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Convolutional+Deep+Belief+Networks+on+CIFAR-10#0`.

[35] M. Choi, Y. Zhang, K. Han, X. Wang, and Z. Liu, "Human Eyes Inspired Recurrent Neural Networks are More Robust Against Adversarial Noises," pp. 1–16, 2022. [Online]. Available: `http://arxiv.org/abs/2206.07282`.

[36] K. Kar, J. Kubilius, K. Schmidt, E. B. Issa, and J. J. DiCarlo, "Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior," *Nature Neuroscience*, vol. 22, no. 6, pp. 974–983, 2019, ISSN: 15461726. DOI: `10.1038/s41593-019-0392-5`.

[37] E. Kim, J. Rego, Y. Watkins, and G. T. Kenyon, "Modeling biological immunity to adversarial examples," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4665–4674, 2020, ISSN: 10636919. DOI: `10.1109/CVPR42600.2020.00472`.

[38] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *10.3. Deep Recurrent Neural Networks — Dive into Deep Learning 1.0.0-alpha0 documentation*, May 2019. [Online]. Available: `https://d2l.ai/chapter_recurrent-modern/deep-rnn.html`.

[39] J. Kubilius, M. Schrimpf, A. Nayebi, D. Bear, D. L. Yamins, and J. J. DiCarlo, "CORnet: Modeling the Neural Mechanisms of Core Object Recognition," *bioRxiv*, no. September, 2018, ISSN: 2692-8205. DOI: `10.1101/408385`.

[40] S. Yan, X. Fang, B. Xiao, H. Rockwell, Y. Zhang, and T. S. Lee, "Recurrent Feedback Improves Feedforward Representations in Deep Neural Networks," pp. 1–10, 2019. [Online]. Available: `http://arxiv.org/abs/1912.10489`.

[41] J. C. Hart, "Perlln noise pixel shaders," *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, no. WORKSHOP, pp. 87–94, 2001. DOI: `10.1145/383507.383531`.