The Relationship Between Image Size and Model Depth In CNNs

Kenneth Hudson

khud1010@gmail.com

Abstract

Convolutional neural networks (CNNs) have been around for decades and have proven themselves to be powerful image classification models. However, many of the default input sizes for CNNS are small images, usually around 256 by 256 pixels. This study will test how image size and model depth are correlated by testing different combinations of the two. The model is made in blocks of convolutional, pooling, and dropout layers. The images are from the National Institutes of Health's chest x-ray dataset online. Medical images were used, instead of a more common dataset like MNIST, to test more complex relationships that may exist between areas of an image. The goal of this study is to find the relationship, if it exists, between image size and model depth.

Keywords:

CNN, Chest X-ray, image-size, model architecture

Table of Contents

Introduction and Problem Statement

The overall goal of this paper is not to exceed some previously obtained validation accuracy but to determine the relationship between the number of layers in a Convolutional Neural Network (CNN) and image size and how their effects on model performance.

The development of CNNs was not only pivotal for the field of deep learning, but also has had a significant impact on society through applications in image and video processing. Models made for image classification tasks have become a trend and competitions, like ImageNet, are numerous. Researchers everywhere are looking to develop models that can outperform the competitor. The focus of this paper is to examine a model's architecture, specifically the depth or number of layers (blocks), and determine if there is an optimal number of layers for certain image sizes while performing a computer vision task. The experiments run on the model will not only look at how different depths affect the model's performance, but also test image sizes to determine if a correlation exists between the performance of deeper models and larger images.

Literature Review

In 1980, Kunihiko Fukushima introduced the Neocognitron (Fukushima 1980), which laid the groundwork for pattern recognition and feature extraction. In this paper, Fukushima relates the architecture to a model that Hubel and Wiesel (Espinosa & Stryker 2012) made after experimenting with cats. The model's architecture utilizes alternating layers of 'simple' and 'complex' cells that cascade to allow for feature extraction and local shift tolerances, respectively. This was the beginning of the convolutional and pooling layers that are used today in modern image recognition.

The next major breakthrough for CNNs would come almost a decade later when John Denker and his team introduced a neural network for recognizing hand-written digits (Denker et. al 1988). The paper goes into detail about the generation of feature maps that check for various features of a number (such as vertical or horizontal lines). As the inputs progress through the layers of the model, more complex features (circles and squares) can be detected. The only downside to this technique was the laborious task of choosing the constraints. This sparked Yann LeCun and his team to release their own paper in 1989 where they introduced the concept of backpropagation to neural networks (LeCun et al. 1998).

LeCun et al used backpropagation on a neural network through a technique they called 'weight sharing'. Each layer of their network shares a common variable that imposes an equality

constraint on each layer connection. At the end of a training cycle, each of these variables, or weights, is updated based on an activation function and learning rate. By having a very small learning rate, the weights of the model could be adjusted slightly. This was a massive breakthrough for deep learning because it meant that the engineers no longer needed to 'hard code' weights into the model for every pass. Models now had the ability to learn from past iterations by minimizing the margin of error and pushing the model to better classify novel inputs.

In 2012, Krizhevsky, Sutskever and Hinton introduced AlexNet, which utilized the rectified linear unit (ReLU) as an activation function and became the first CNN to win the ImageNet classification challenge (Krizhevsky, Sutskever & Hinton 2017). This was a shift away from the sigmoid or tanh activation functions, which were seen as standard at the time. The ReLU activation function forces a minimum of zero so no negative values are possible for the output. Since the release of AlexNet, CNN's have dominated the ImageNet contest and newer generations of these models have begun to rise in prominence. These have paved the way beyond simple CNNs and into many different deep learning architectures.

This paper will utilize some of these previous works in order to build a custom CNN neural network and use it to train multiple models on combinations of image sizes and model depth. Similar work has been performed by Wang et al. (2017) in their paper featured on the National Institutes of Health's website, where the dataset is found. In the paper, they describe the construction of the dataset, which used radiology reports and various natural language processing methods to detect the various pathologies found in the dataset. Different types of model architectures, including the previously mentioned AlexNet (Krizhevsky, Sutskever & Hinton 2017), were then trained on the data in order to solve the multi-label classification problem.

Others have put out papers that seek to use different model architectures in order to achieve better results on this same dataset. Li Yao's team attempted to use a DenseNet (Huang et al. 2017) encoder and then a LSTM (Long Short Term Memory) decoder in order to classify these same chest x-rays (Yao et al. 2018). Kufel et al. (2023) decided to use EfficientNet (Tan & Le 2020) and found great success when compared to other models. These works provide great insight into how different architectures can be used to train models more efficiently and achieve better validation results. However, they do not try to uncover the relationship between the number of layers in a model and the image resolution it is trained on.

Data

Chest X-Rays are a common tool used in the medical field to determine if a patient is suffering from any complications related to some of the body's most vital organs, the heart and

lungs. Much of a diagnosis, and the subsequent treatment, relies on the proper interpretation of the x-ray by a physician. The data used to train these models during experimentation were supplied by the National Institutes of Health as an open source dataset (a link can be found in the Data Availability section). Each image in the dataset is originally a 1024 x 1024 pixel .png file in black and white. The dataset also comes with a .csv file that contains the labels for each image. There are 15 total labels in the dataset and some images had multiple labels assigned to it. The label 'No Finding' represents more than 50% of the data (Figure 1), while infiltrations, atelectasis and effusions make up more than 25% of the images that do have labels associated with them. Due to this class imbalance, these labels were filtered into two classes each representing almost half of the entire dataset. By sorting the labels into binary dataset, not only is the data more balanced (Figure 2), but the model will train on a simpler classification task, instead of a multi-label classification problem. It will determine whether or not the x-ray has 'No Findings' or should be flagged as 'Unhealthy'. The images were then separated into training and validation folders to be used for their respective tasks based on two text files accompanying the dataset. This split cut the full dataset of 112,120 images into 86,524 (77.17%) training images and 25,596 (22.83%) validation images.

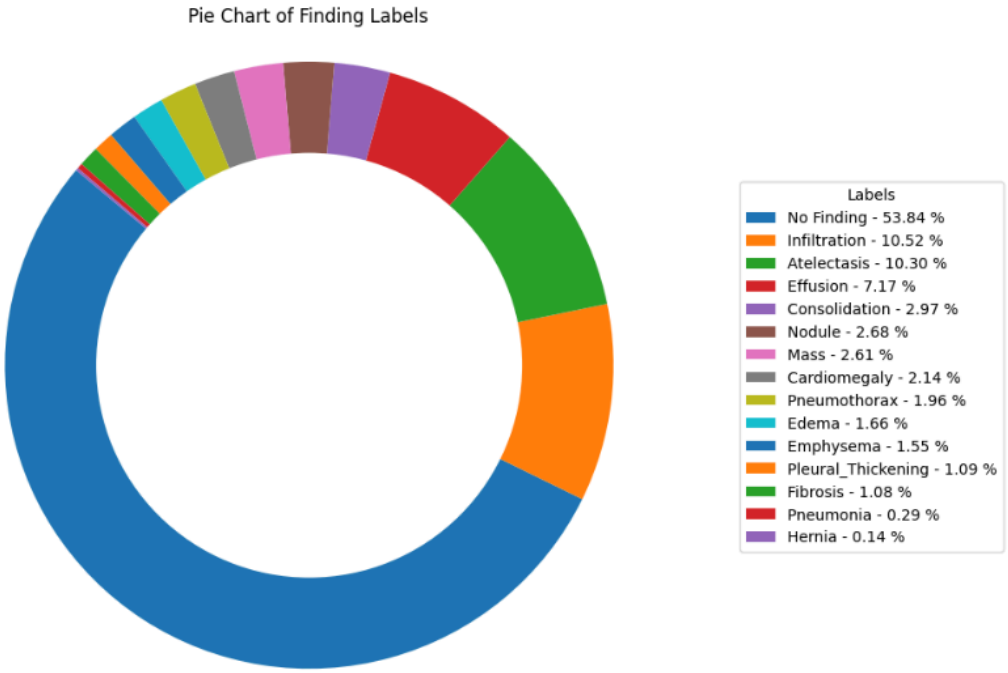


Figure 1: Pie chart showing the distribution of labels (based on percentage) in the dataset

Class	Count	Percentage
No Finding	60361	53.84%

Unhealthy	51759	46.16%
Total	112120	100%

Figure 2: Number of images in each class along with the respective percentage of images in the total dataset

Methods

The CNN architecture that was used for this model consists of a 'block' of convolution, batch normalization, max-pooling, and dropout layers, followed by a flattening layer, and finally two dense layers (Figure 3). The final dense layer has a sigmoid activation function resulting in a binary output of either healthy or unhealthy lungs. The number of blocks was increased sequentially from one block to four and trained on the same image data (Figure 4). As the number of blocks increased, so did the filter sizes for each convolutional layer (much like Fukushima's Neocognitron mentioned previously). This increase in filter size was done using the product of the original filter size and the number of CNN blocks in the architecture. After this was done, the image size was increased and the same procedure of training was performed on each of the following image sizes (in pixels): 128x128, 256x256, 512x512, 1024x1024. After training was complete a total of 15 models were trained. Errors arose due to dimensionality constraints during the training of 128x128 images with 4 CNN blocks. After the third cnn block, the output sizes are too small to be sent to a fourth block and thus, training stops. For this reason, 128x128 images were not trained on the 4 block model.

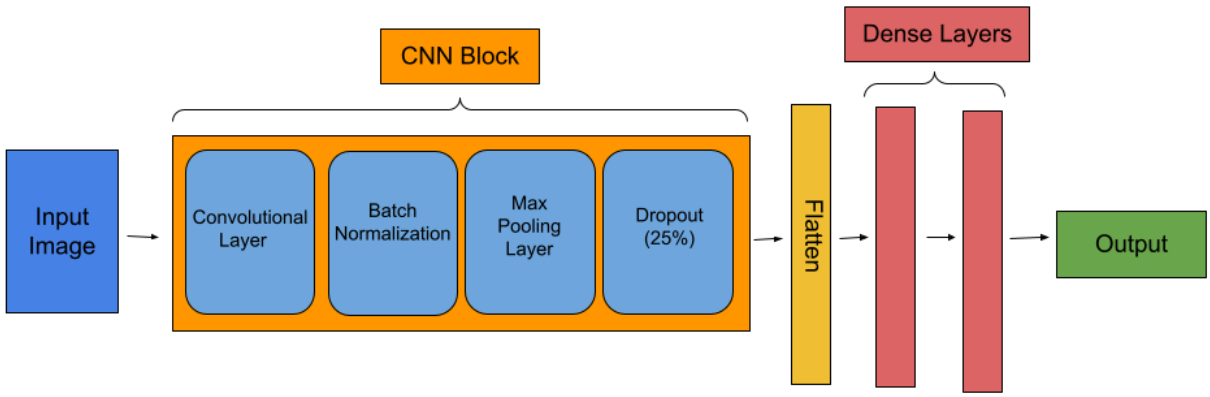


Figure 3: CNN architecture for first run of every image size

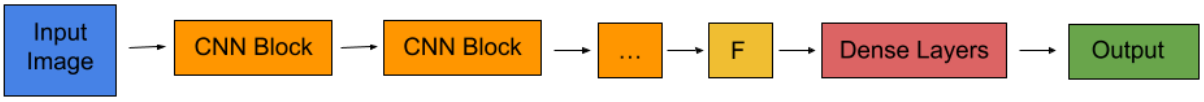


Figure 4: CNN blocks in sequential order

Training was completed using Keras and Tensorflow libraries as well as a custom function that added blocks as deeper networks were tested. This function also included the filter size increase based on the number of blocks in the architecture. Each training run had 10 epochs and the Adam optimizer was used with a learning rate of .0001. The loss function used during training was binary cross entropy, which is suitable for a binary dataset. Batch sizes for the 128x128 and 256x256 image sizes were increased in order to speed up the run time of the python script. Training all of the models took just over 13 hours, with a large portion of that time being taken up by the 1024x1024 image size training.

Results

As stated above, 15 total models were trained on the same data with the same hyper parameters. Each model had a combination of image size and number of CNN blocks. The initial results are difficult to truly analyze as the model did not appear to train and learn the relationships that are needed in order to make any meaningful inference on new data. Some models had more success during training but almost all of them failed to achieve above a 50% validation accuracy. The accuracy and loss plots for every model can be found in the Appendix but will be analyzed here. In the plots, 'Layers' refers to the number of additional CNN blocks in each architecture after the first one. So if there are zero layers, that means it is the basic architecture seen in Figure 3 (above) and for each layer, an additional block is added like in Figure 4. The results will be presented by the number of blocks below:

One Block

For the smaller image sizes, training metrics trend in the right direction, accuracy goes up and loss goes down. However, when looking at the validation metrics, it is clear that the model did not generalize the data as the accuracy and loss remain relatively stagnant. In larger image sizes (512x512 and 1024x1024), validation loss increases towards the end of training and validation accuracy remains the same. The smallest image size achieved the best validation accuracy of 52.46%, although it remained stagnant throughout the epochs. When compared to others, these models overall seemed to have the best performance across all image sizes.

Two Blocks

The addition of another convolutional block gave similar results for the two larger image sizes but saw a similar validation accuracy (53.61%) for the 256x256 image size as was seen in the 128x128 image size with one CNN block. The smaller image sizes still saw an increase in loss in later epochs, although not as dramatic as the larger image sizes. The 256x256 image size model did see improvements to both metrics during training. Larger images saw little variation during training to either metric but saw a large increase in loss during later epochs.

Three Blocks

A third CNN block brought about changes during training that affected all image sizes. As stated before, the validation metrics either remain stagnant or get worse at later epochs. However, the training metrics show that smaller image sizes had more fluctuation than previously seen in smaller architectures. Training metrics for 128x128 spike up and down but generally trend in the correct direction. Whereas, larger image sizes appear to have a smoother slope to their metric graphs. In all cases, training metrics appeared to be trending towards a trained model but the validation accuracy shows that is not the case.

Four Blocks

Finally, the fourth block was added to the model architecture and showed that smaller image sizes did not tolerate the multiple convolutions well. 256x256 images saw little movement in both training metrics and validation metrics and was the only training accuracy of all the models to end below 50% . It also ends with a validation accuracy higher than the training accuracy, which does not happen in any other model. The larger image sizes looked to train similarly to the three block architecture, although it performed slightly worse in terms of training accuracy overall.

Discussion

The results from training are difficult to analyze without further work on the model pipeline. A custom data loader was used in order to initialize and load the images in batches to the model for training. The suspicion most likely rests on something in this pipeline having an effect on what data was given to the model during the validation of each epoch. Using the training metrics, it was seen that smaller image sizes worked best with few convolutions and sometimes not at all as the model depth increased. On the contrary, larger image sizes did not have much variation in training metrics with fewer blocks. When comparing the 128x128 + 1 CNN block architecture versus the 1024x1024 + 4 block architecture, the metric plots are almost identical. In each graph, accuracy is seen to steadily rise and loss appears to also have a steady

decline. Without proper validation metrics, these findings cannot be confirmed at this time but the data suggests that adding convolutions to larger images helps the model to reduce the dimensionality enough to understand certain relationships in the image data.

Conclusions

The work done in this paper shows that there could be a correlation between image size and network depth during training. However, as previously stated, the validation metrics for all models trained are unreliable and thus these findings cannot be proven yet. Smaller image sizes suffered more when given to larger networks due to the highly reduced dimensionality that is eventually given to the dense layers at the end of a training loop. If the output shapes from each previous layer of a CNN block get smaller and smaller, then there is eventually no more information in the input shape given to the dense layers. The larger images did not need to worry about drastic dimensionality reductions and thus performed well on deeper networks while smaller images did not.

This basic CNN approach showed that it is not always better to add more depth to models and that the additional layers might hurt more than help a model. Image sizes during training do suggest that smaller images can still achieve similar accuracies as larger images. Although deeper networks are only useful when training large images that can handle multiple dimensionality reductions. The training of these models took significant resources, especially for larger images and more work will be dedicated to tweaking hyperparameters and ensuring the data pipeline works as intended.

Directions for Future Work

The machine used for this project was a personal home computer, which limited the speed at which the larger image sizes could be run. So in the future, it would make sense to rent a virtual machine through one of the many online sources. Model training was also very challenging because multiple models were trained using the same hyper parameters. This was done in order to prevent conflicting variables skewing the results, but ultimately different hyper parameters are needed for different tasks. There is also the possibility of tweaking the learning rate using a couple different methods. Running a learning rate finder could help find the optimal rate or changes the learning rate could be made during training using a scheduler. More attempts will be made in the future to optimize the hyper parameters of the model and data loader to make sure each model is performing at its best possible level. Also, a more powerful machine will hopefully be used to speed up training and allow for more opportunities to test different hyperparameters.

Data Availability

National Health Institutes Chest X-Ray Dataset:

https://nihcc.app.box.com/v/ChestXray-NIHCC

Code Availability

Github

https://github.com/khud1010/CNN-Depth-vs-Image-Size

References

Denker, J, Gardner, W., Graf, H., Henderson, D., Howard, R., Hubbard, W., Jackel, L., Baird, H., & Guyon, I. 1988. "Neural Network Recognizer for Hand-Written Zip Code Digits." *Advances in Neural Information Processing Systems*. Morgan-Kaufmann.

Espinosa, J. Sebastian, and Michael P. Stryker. 2012. "Development and Plasticity of the Primary Visual Cortex." *NCBI*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3612584/.

Fukushima, Kunihiko 1980. "Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." *Biological Cybernetics*, *36*(4), 193–202. https://doi.org/10.1007/bf00344251

Gao Huang, Zhuang Liu, Laurens Van Der Maaten and Kilian Q. Weinberger. 2017. "Densely Connected Convolutional Networks." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2017.243.

Krizhevsky, Alex, Sutskever, Ilya, & Hinton, Geoffrey 2017. ImageNet classification with deep convolutional neural networks. In *Communications of the ACM* (Vol. 60, Issue 6, pp. 84–90). ACM. https://doi.org/10.1145/3065386

Kufel, Jakub, Bielówka, Michal, Rojek, Marcin, Mitręga, Adam, Lewandowski, Piotr, Cebula, Maciej, Krawczyk, Dariusz, Bielówka, Marta, Kondoł, Dominika, Bargieł-Łączek, Katarzyna, Paszkiewicz, Iga, Czogalik, Łukasz, Kaczyńska, Dominika, Wocław, Aleksandra, Gruszczyńska, Katarzyna, & Nawrat, Zbigniew 2023. "Multi-Label Classification of Chest X-ray Abnormalities Using Transfer Learning Techniques." *Journal of Personalized Medicine, 13(10), 1426-*. https://doi.org/10.3390/jpm13101426

LeCun, Yann, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. “Gradient-Based Learning Applied to Document Recognition.” *Stanford Vision Lab.* http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf.

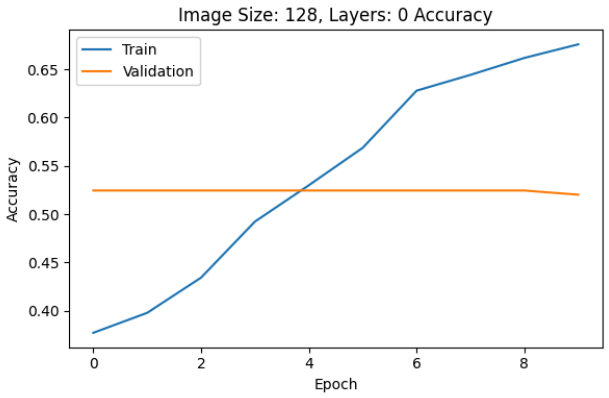
Tan, Mingxing, & Le, Quoc V. 2020. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” *arXiv*. https://doi.org/10.48550/arxiv.1905.11946

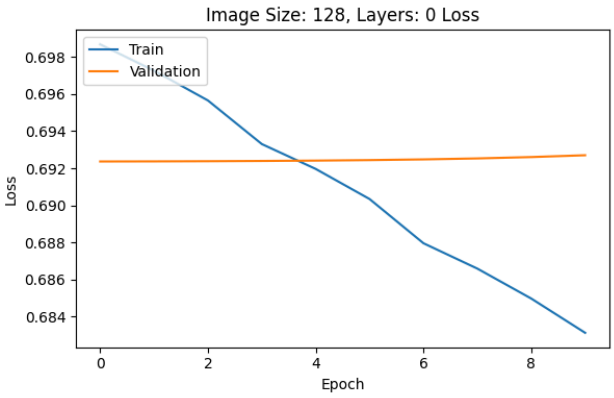
Yao, Li, Poblenz, Eric, Dagunts, Dmitry, Covington, Ben, Bernard, Devon, & Lyman, Kevin 2018. “Learning to diagnose from scratch by exploiting dependencies among labels.” *Cornell University*. https://doi.org/10.48550/arxiv.1710.10501

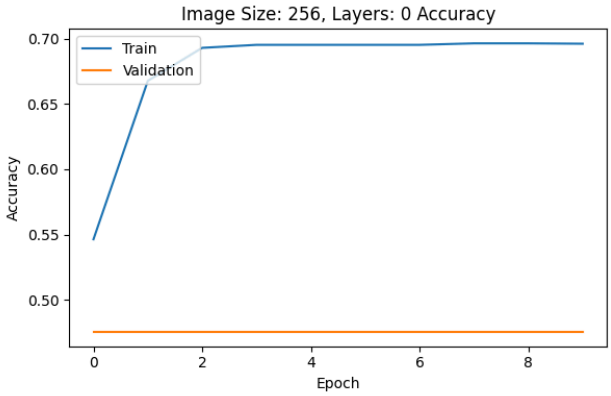
Appendix A

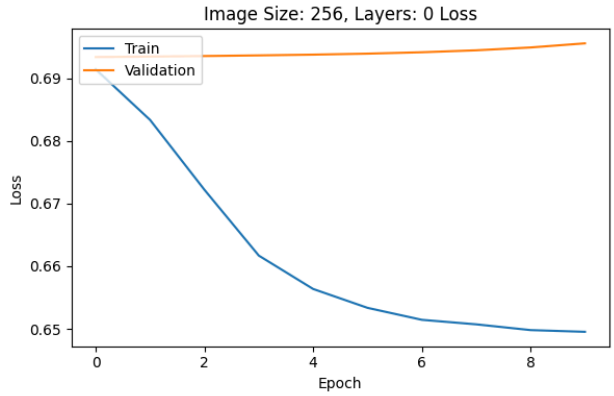
Accuracy and Loss Plots for all 15 models

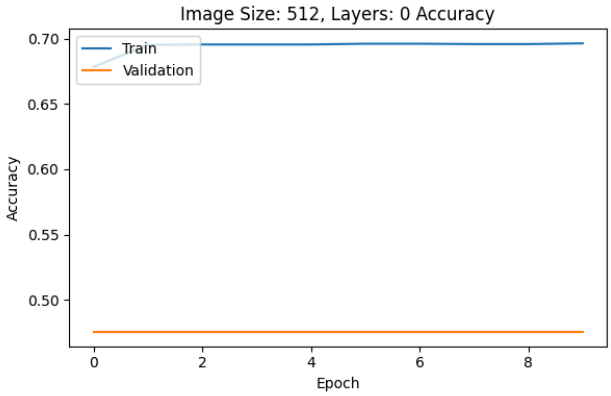
One Block:

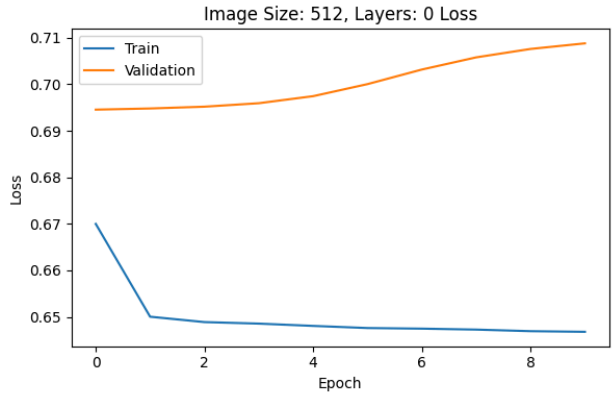












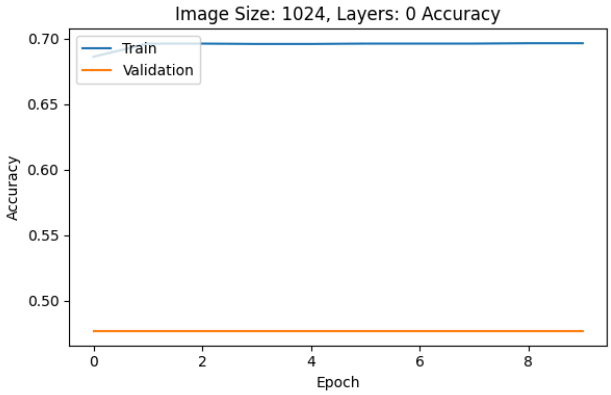


Image Size: 1024, Layers: 0 Accuracy
Train
Validation
Accuracy
Epoch

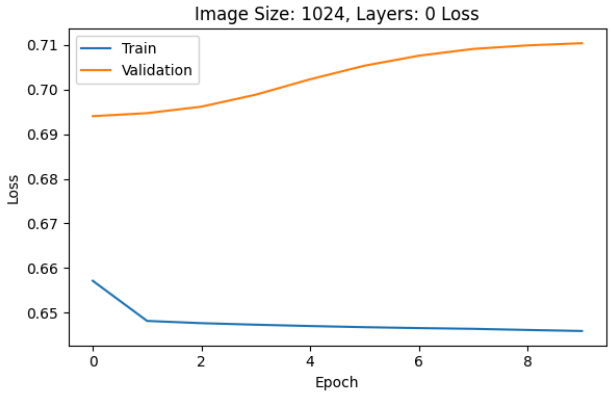


Image Size: 1024, Layers: 0 Loss
Train
Validation
Loss
Epoch

Two Blocks:

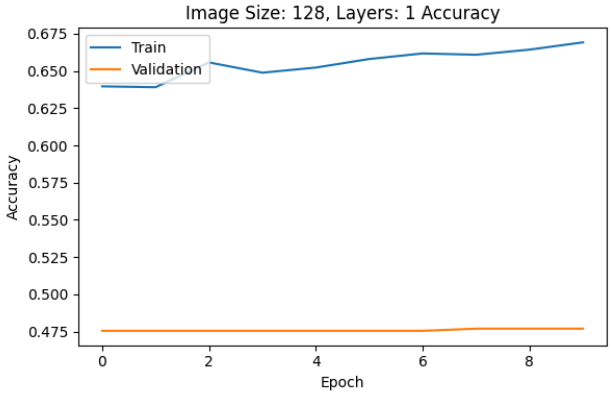


Image Size: 128, Layers: 1 Accuracy
Train
Validation
Accuracy
Epoch

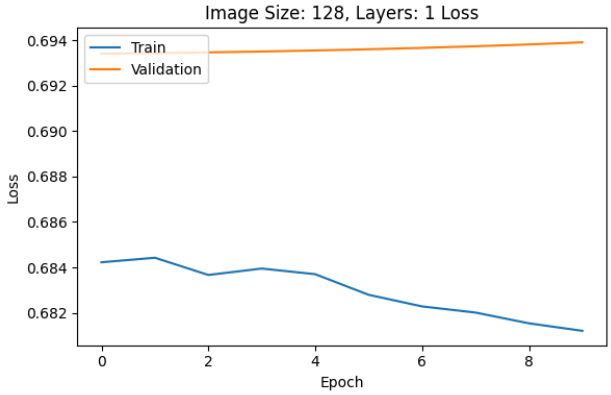


Image Size: 128, Layers: 1 Loss
Train
Validation
Loss
Epoch

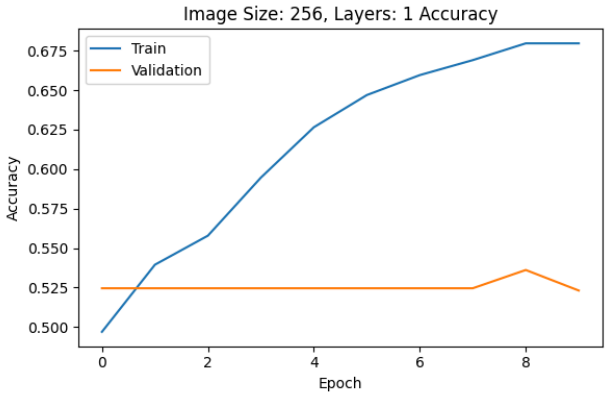


Image Size: 256, Layers: 1 Accuracy
Train
Validation
Accuracy
Epoch

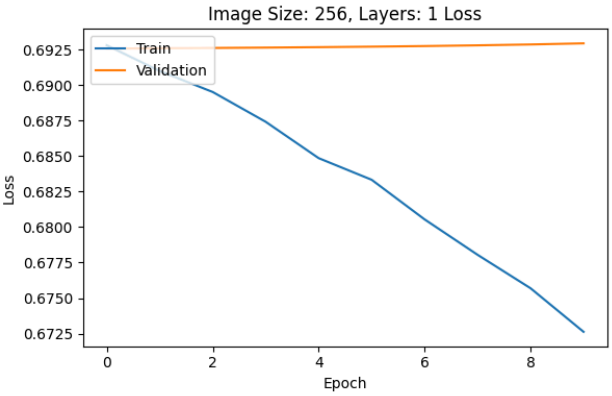
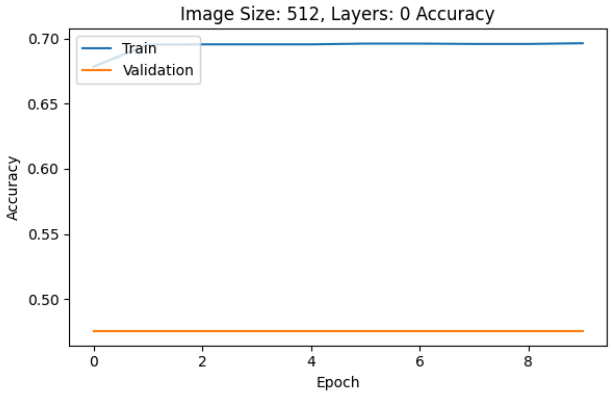
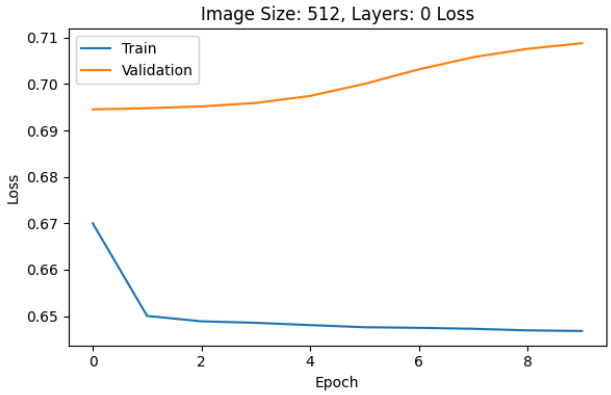
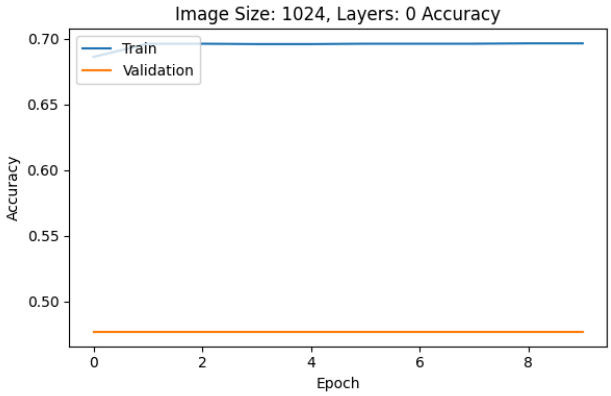
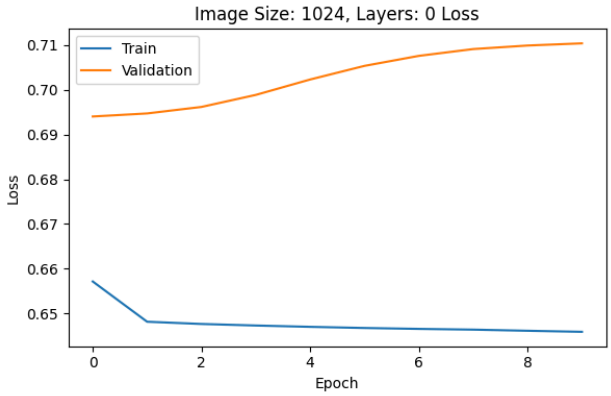


Image Size: 256, Layers: 1 Loss
Train
Validation
Loss
Epoch

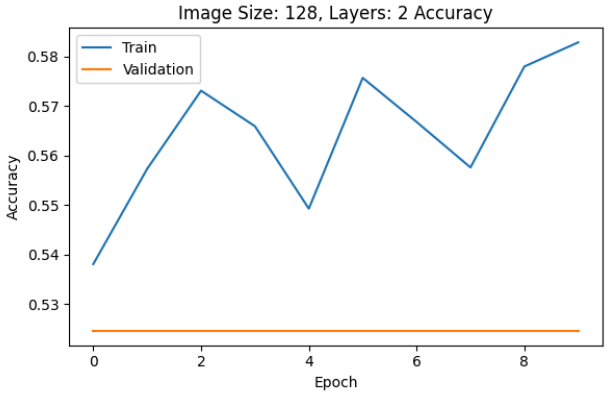


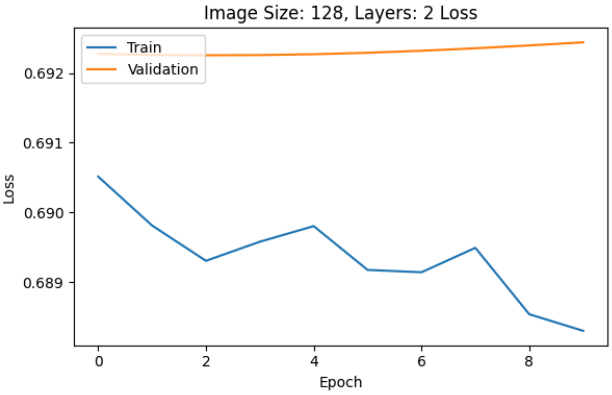


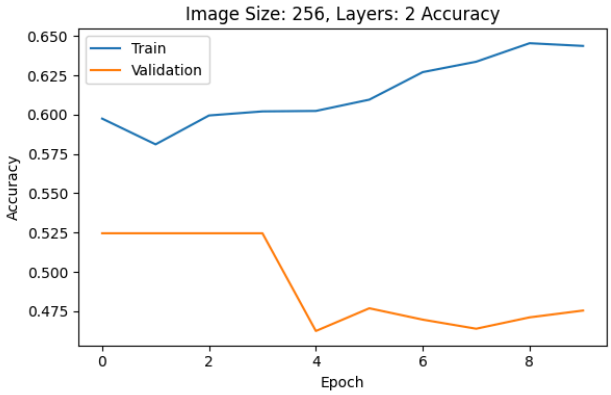


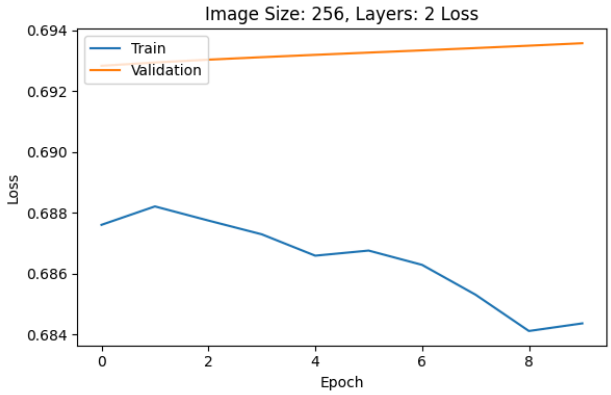


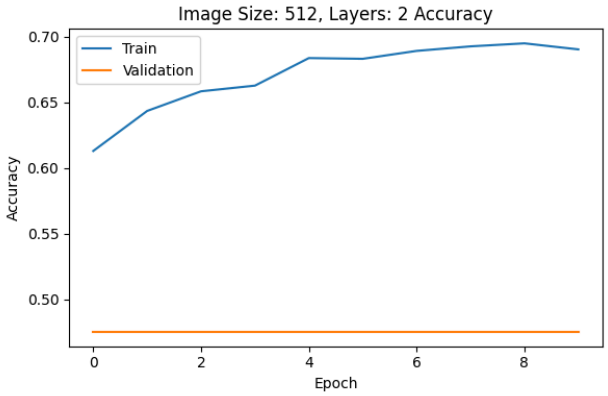
Three Blocks:

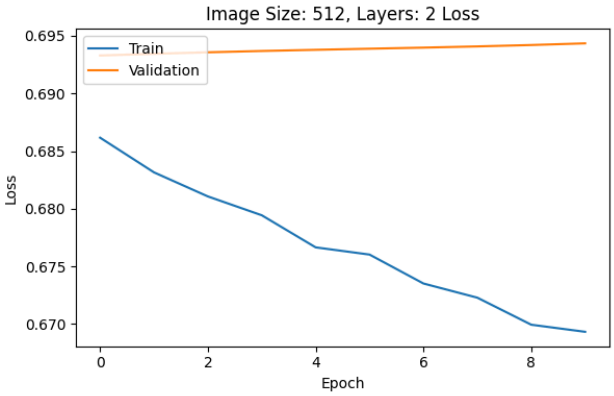


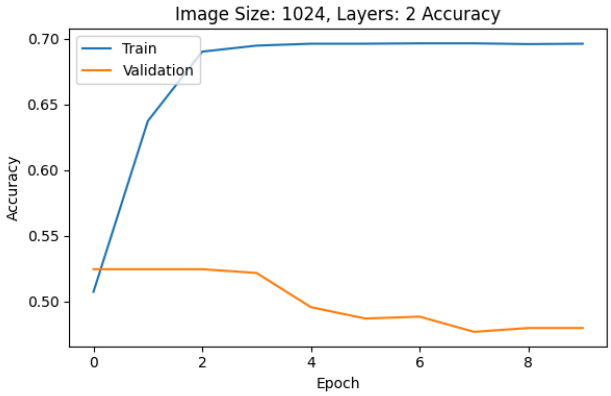


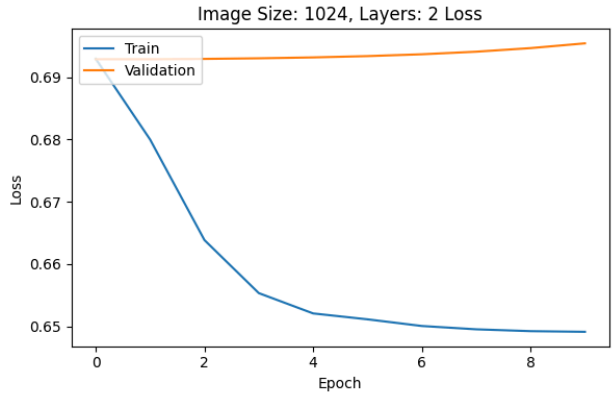












Four Blocks:
(128x128 excluded)

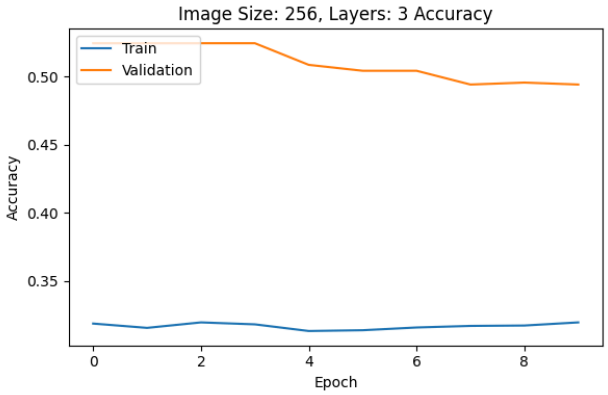


Image Size: 256, Layers: 3 Accuracy
Train
Validation
Accuracy
Epoch

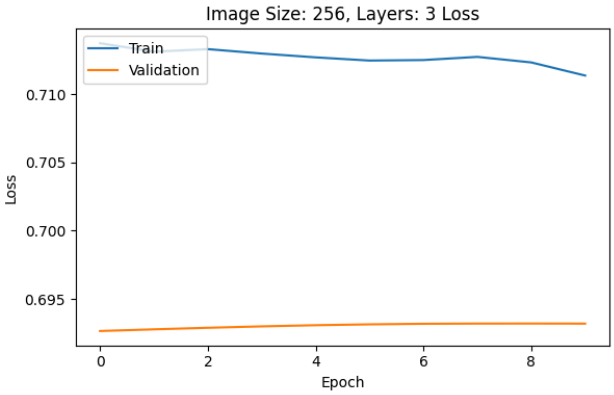


Image Size: 256, Layers: 3 Loss
Train
Validation
Loss
Epoch

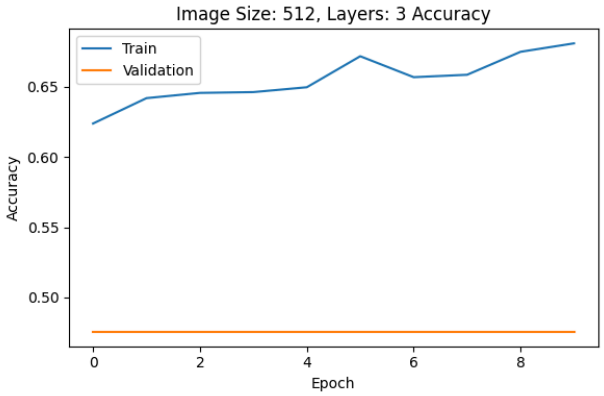


Image Size: 512, Layers: 3 Accuracy
Train
Validation
Accuracy
Epoch

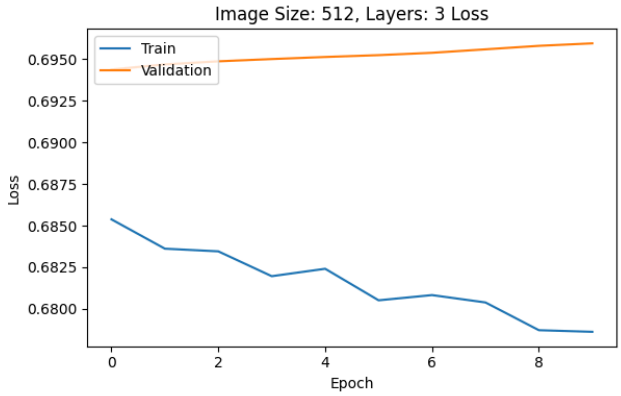


Image Size: 512, Layers: 3 Loss
Train
Validation
Loss
Epoch

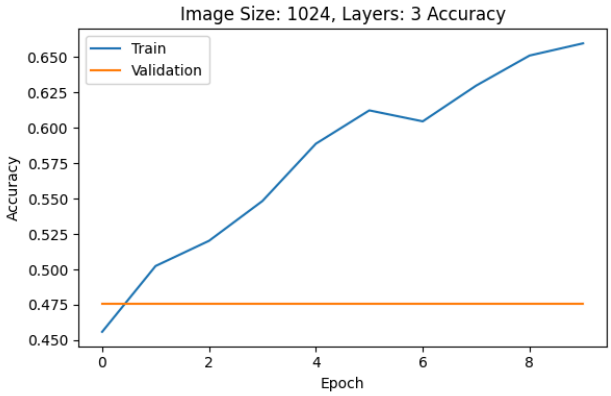


Image Size: 1024, Layers: 3 Accuracy
Train
Validation
Accuracy
Epoch

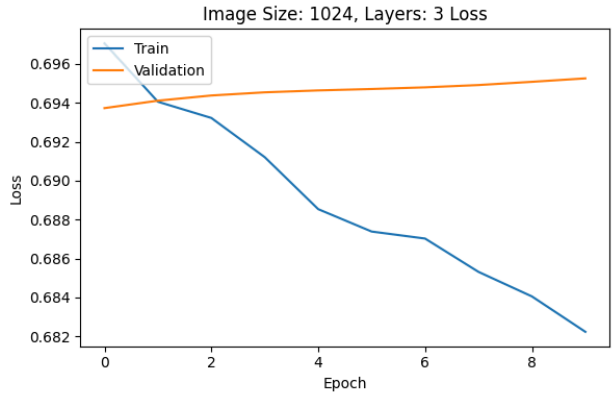


Image Size: 1024, Layers: 3 Loss
Train
Validation
Loss
Epoch