DAT341: Assignment 5

Johannes Berger
Personal number: 000917
johberge@chalmers.se

Cecilia Nyberg
Personal number: 990106
cecnyb@chalmers.se

Elin Stiebe
Personal number: 000210
elinsti@chalmers.se

Abstract

This paper aims to find an optimal model to classify melanoma by using image recognition. The model was found by evaluating several approaches in isolation and later some in combination. The different methods used were augmenting the training data, normalizing, residual connections, and using a pre-trained model. The best-performing model found used transfer learning with a pre-trained model. This model reached an accuracy of 0.88 and was achieved by re-training a ConvNeXt model.

1 Introduction

Skin cancer is one of the world's most common cancer types (Venosa, 2018), though, it is very curable if it is caught early. Melanoma is *localized* in its early stages (Venosa, 2018). This means the cancer is not spread in the body but is situated on the skin. Queues to reach a medical professional can be long, and introducing ML into the diagnosis of skin cancer could be one way of reducing the work hours for practitioners in healthcare. With this background, developing an ML model that can detect melanoma becomes highly interesting.

How would such a model be designed? And what aspects are important to consider to build an efficient model? These questions are broken down into more detailed subquestions: How well does a basic convolutionary neural network perform in recognizing melanoma in images, and how does the integration of data augmentation, batch normalization, and residual connections affect this performance? Additionally, the performance will be compared to a model using transfer learning with pre-trained models.

Further, what are some drawbacks or limitations to ML models in diagnosing melanoma, and are there biases built into the data and model that could have serious effects if the model were deployed? We aim to provide answers to these questions in the following sections.

2 Technical Solutions

As a first step, a baseline model was used for classifying the images. At this point, the data was normalized and turned into tensor format but not further transformed. It was also checked that the data was balanced which it was. 10 epochs were run with a batch size of 128. A few alternatives were tried at random to decide which layers should be used. This proved harder than expected, and the model struggled to find any meaningful patterns in the data. The output was only zeros every time, consequently, the validation accuracy was always 0.5 and the loss did not decrease. Therefore, inspiration was taken from the article (Shepherd, 2021) about using image recognition for skin cancer detection, and a similar model architecture as used in the article was implemented. A difference for the layers in (Shepherd, 2021) compared to what was initially tried out was that they used leaky ReLU instead of ordinary ReLU and also a larger Kernel size (5) for the convolutional layers. The leaky ReLU multiplies negative values with a small value instead of making them zero, preventing the "dying ReLU problem", (Shepherd, 2021), which is that gradients become zero and are not updated after that point, resulting in large parts of the network being dead. A consequence of this can be that the model always outputs zeros. Furthermore, a larger Kernel size is better at finding global features and larger patterns in the image, and a small is better at noticing small details (Devron). The updated model performed better and was therefore selected as the baseline model. The model consists of four blocks and two heads. The layers for this baseline model can be seen in appendix A.

Several additional techniques were added to the baseline. First tried in isolation and then in some combinations of techniques to try and find the highest performing model. The first technique added was a normalization technique. The batch normalization was used and added after each convolutional layer as seen in appendix C. Batch normalization is most efficient on larger batches as it normalizes over it, so could be a suitable normalization technique when using batch size 128. Further, the data was shuffled to avoid any questions about the randomness within each batch size.

In the next version, a few transformations were used on the data to make the model more robust. The transformations in table 1 were applied to the data. The mean *a* in the normalization was set to a = [0.485, 0.456, 0.406], and the std *b* was set to b=[0.229, 0.224, 0.225]. All of the transformations were added to increase the robustness and prevent overfitting by training the model on images that were skewed in one way or another.

Index	**Transformation**
1	RandomHorizontalFlip(p=0.5)
2	RandomVerticalFlip(p=0.5)
3	RandomAffine(20)
4	Normalize(mean=a, std=b)

Table 1: Applied transformations for the custom model.

Additionally, residual connections were added to the baseline model. When using residual connections, input going through layers is added with input passing by. This prevents information loss in later blocks of the model, which can otherwise occur when the input has been processed a lot. The residual connections were added after each of the four blocks in the model. To be able to add the processed tensor with the skip tensor, these two have to be the same size. This can either be done by having the same size for input and output, or by re-sizing the tensor passing by. Each approach was tried out to see which would work the best. Using the same input and output size created some restrictions on the model, so the baseline model had to be refined, and the parameters changed. The layers for both strategies can be seen in appendix D.

As a last individual strategy to try, transfer learning was used by using a VGG16- and ConvNeXt-based model for feature extraction and classification. Two common approaches in transfer learning are fine-tuning a pre-trained model and using a pre-trained model for feature extraction.

Inspiration was taken from (Simonyan and Zisserman, 2015) as the VGG16 image classification model was adapted and fine-tuned with batch normalization. It includes a feature extractor consisting of 16 convolutional layers and a three-layer feedforward neural network as a classifier. Because the model is typically evaluated on the ImageNet 1k dataset, the last layer was changed to have a single output for binary classification. Since this last layer is initialized randomly, at least these weights have to be re-trained. Furthermore, a model-specific data augmentation was employed to maintain the integrity of the input data's representation as expected by the model.

Another attempt was made to train additional layers of the classifier. The hypothesis was that this might improve the classification accuracy, as more weights are adopted to the data.

Lastly, a few methods were run, combining several of the options described above. Two of the combinations are described in this paper.

The data that the models were trained and evaluated on were derived from the most extensive challenge in skin cancer classification as outlined in (Codella et al., 2019). The used evaluation metrics were loss and accuracy, the results of these two metrics can be seen in all of the tables describing each of the training epochs. The evaluation is discussed further in section 5 about ethical considerations.

3 Experiments/Results

The next sections describe the results of each model on the data. Section 3.1 presents the baseline, section 3.2 the results with normalization, section 3.4 focuses on the use of residuals while section 3.3 is results when using random transformations on the data. Section 3.5 shows the results of using a pre-trained model.

3.1 Baseline Model

The result when using the baseline model described in appendix A can be seen in appendix B in figure 2. The baseline model reached a validation accuracy of 0.75 after ten epochs and a loss of 0.47.

3.2 Using normalization

Batch normalization was used to normalize the input in the model. The results are in appendix E in figure 3. The results show a higher validation accuracy (0.81) and a lower loss (0.32) than the baseline model in section 3.1. It shows that the use of normalization positively impacts the performance and makes it easier for the model to find meaningful patterns in the data.

3.3 Data Augmentation

The layers from the baseline models, shown in appendix A, were used, together with the transformations displayed in section 2 in table 1. Figure 6 in appendix G shows the results of applying the transformations to the data. The validation accuracy of 0.75 and loss of 0.45 is similar to that of the baseline model in section 3.1. Using transformations did not really affect the results. Transformations usually result in a more robust and generalized model, so it might have an impact on how the model performs on data that is more diverged than the data it was used on. Without testing this it is not possible to say though. The reason for the model not performing better with transformation could be tied to the types of transformations chosen or the images' nature. All of the birth-mark images were taken in quite a standard way, meaning there is no significant advantage of being able to classify a skin mark that is turned to a certain degree, etc. Removing overlaying artifacts, such as hair, in further pre-processing could be another useful data augmentation approach, but was not investigated in this paper.

3.4 Residual connections

Residual connections mitigate several issues in deep neural nets, such as information loss or exploding or vanishing gradients (Wong, 2021). Residual connections were used in two different ways. The result of having the same input and output size is shown in appendix F in figure 4, and the result of re-shaping the tensors is shown in 5. The validation accuracy was 0.79 for both versions. However, the results show that re-shaping the tensors was more efficient at reducing the loss (0.36 for re-shaped vs 0.41 for using the same size). This could be because it was easier for the model to identify important features when looking at the input at different scales. Additionally, when re-shaping the tensors instead of having the same input and output size, fewer parameters could be used which reduced computational complexity. Overall, the residual connections result in the model performing better in all aspects, for both approaches, indicating that the baseline model might have had issues with information loss or exploding/vanishing gradients.

3.5 Transfer Learning

Lastly, some forms of transfer learning were used to approach this classification task. The finding was that the training was more stable, the fewer layers we trained, but the overall accuracy remained mostly similar. Training only the last layer and, to a lesser extent, the last two layers, seems to provide a balance between leveraging the pre-trained model's knowledge and adapting to the new task. Especially when updating pre-trained weights, one has to be careful to preserve the intrinsic knowledge of the model, and there was an occasional drop in accuracy during training which could be attributed to this effect. A lower learning rate consistently yielded more stable training processes and better results, at the expense of slower overall training progress. One interesting approach might be to use a higher learning rate for the last layer to build up knowledge from scratch more quickly and a lower one for the pre-trained classifier layers to preserve knowledge. Overall, the challenge in fine-tuning remains to find a balance between many parameters, the mutual dependence of which is not obvious.

Additionally, an investigation was made into a more recent and state-of-the-art model family called ConvNeXt (Liu et al., 2022), which is conceptionally quite similar to the custom implementation of this paper, as it is primarily based on convolutional neural networks. Additionally, it showed a decent balance between top-1 accuracy on ImageNet 1k and remaining rather small.

The computational cost had to be considered with every operation on, and with, the rather large pre-trained models. Training at times roughly 120m parameters requires significant compute power, and we relied on supercomputer resources with powerful GPUs to achieve short training durations.

In conclusion, the VGG and ConvNeXt-based models achieved validation accuracies of 0.88 and 0.89, respectively, outperforming the custom models by a noticeable margin.

3.6 Combination of Strategies

Two of the combinations tried were (i) using residual blocks and batch normalization, and (ii) using residual blocks, batch normalization, and transformation. In these combinations, the residual tensors are reshaped since this approach proved most beneficial in isolation in section 3.4. The layers of the two combinations are consistent with those in appendix C though the second version also uses the transformations described in section 2 in table 1. Figure 7 in appendix H displays the residual blocks and batch normalization results. It reached a validation accuracy of 0.84 which is higher than for the baseline and most of the other previously tried setups. The loss was 0.2 which is also lower than the baseline. Batch normalization should usually reduce overfitting and make the model more robust. The combination of using this and residual connections proved beneficial for the model.

Figure 8 in appendix I shows the results from the second combination. The validation accuracy reached 0.83 in its 10th epoch and a loss of 0.33. Again, using transformations did not yield a higher accuracy than not using them. As previously stated, it might be because the data it validates on is quite standardized.

Even though the model also using transformations resulted in a similar accuracy, the model without transformations got a lower loss. This is reasonable since the data without transformations is more similar and, therefore, easier to find patterns and specialize in. When the transformation is combined with residual connections, the model becomes more robust and less specialized. In many cases, having a robust and general model is desirable. However, there are also cases when it is not preferred. For example, if all images are taken similarly, with the same angle and rotation, the model could potentially reach higher accuracy when trained and used on well-structured and similar data. There is a trade-off between generalization and specialization, and which one to prioritize depends on the situation. Medical images of birthmarks are taken in a controlled environment, where light, angle, and distance are easily kept in a standard way. Therefore, there is reason to aim for a specialized model and specialized data. However, with the current data, the more general model performed similarly to the more specialized.

3.7 Best performing model

The best-performing model used ConvNeXt-based transfer learning. This model was evaluated on the test set data. The ConvNeXt-based model achieved a test set accuracy of 0.88.

4 Limitations

First, limitations in the performance of the model. The models described above are constricted to the same training data and thus limited in the patterns they might identify. This issue is prominent since it is difficult to get large amounts of patient data, and data is important for ML models (Zhang et al., 2022).

The next limitation of the model is the time it was developed. It takes time to train deep neural models, thus there are most probably several models that perform better than the one found in this paper, though due to the training time, the development of the model was limited to a specific number of runs and evaluations. A connected issue is the limited number of epochs. All runs only did 10 epochs which could have been increased given more time.

Another aspect that the limited time imposed was non-exhaustive hyperparameter tuning of all models. Another such aspect is that only one instance per model was trained, meaning the models could have ended up in suboptimal local minima and thus missed much of their potential. This could have been improved by training the same model several times with different weight initialization.

Now to the limitations in deployment. The model is limited in its ability to explain classifications. The models described in the previous sections are all without any aspects of interpretability. The models are limited in their explainability to users of it.

There are also limitations regarding the evaluation of the models. Only loss and accuracy are considered, but when classifying images as having or not having cancer, only considering these factors can be misleading and unfair to some models. Other evaluation techniques, such as false negatives could be important to consider as well, since misclassifying images with melanoma as healthy can be considered worse than misclassifying images without as sick. More on this in section 5.

5 Ethical Considerations

One important consideration is how the model is evaluated. Throughout this report, the accuracy score of the entire set is used, though the model might have different accuracy scores depending on the skin tone the image is depicted. Therefore it would have been insightful to extend the accuracy score with scores evaluating the accuracy on different skin colours. Another point to consider regarding evaluation is *sensitivity* and *specificity*. The two metrics are often in a trade-off relationship, and thus, developers must choose which to prioritize. In the question of cancer, the question becomes, should the algorithm misclassify patients with cancer or miss instances of the disease? If the disease is missed, people will go untreated, and this is incredibly dangerous according to (Venosa, 2018) yet if patients are misclassified with cancer, they will suffer unnecessary stress and an increased cost for the health care system. These questions must be balanced carefully.

Further, such a classification algorithm can only distinguish between *melanoma* and *nevus* and cannot identify other irregularities that might indicate other diseases. This is something where a clinician can be superior to the algorithm.

Now to the training data. The most optimal would be to get specific stats on the images and decide if all categories of skin tones are included and represented to a good extent. This inspection would demand an ML algorithm on its own. Instead, 50 images from the training set are printed to inspect the skin color around the birthmarks. Out of the 50 randomly printed images, without certainty, one seems to represent a person with dark or darker skin color (by manual inspection). One such print of 50 images is shown in figure 1. Such a lack of representation means that the ML model could perform worse on some groups of people.

Another ethical consideration connected to figure 1 relates to the quality of each image. Some images have a better quality than others. This might depend on the hospital's resources, the clinician's expertise, or the clinic's surrounding environment, such as lighting. Images might be prone to get a certain classification due to other factors in the image. One such example is the black surroundings found in six of the images in figure 1.

A third issue related to the images is related to integrity and privacy rights. Who should have access to images from health care, and how should they be connected to the patients from whom the images are taken? Collecting a massive amount of images is favorable for a successful and high-performing ML model, yet the desire for increased performance can lead to people crossing the line in terms of patient privacy. It is important to consider which images to save and what contracts to draft with patients to make it legal and ethical.

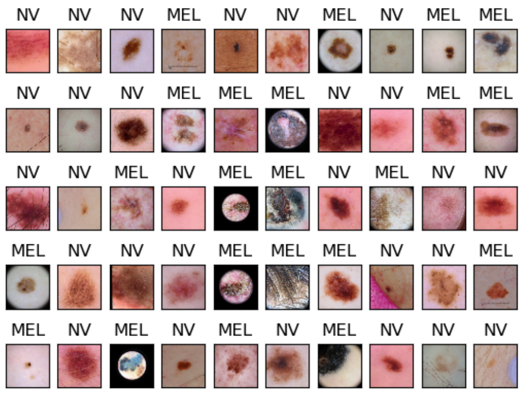

NV NV NV MEL NV NV MEL NV MEL MEL
NV NV NV MEL MEL MEL NV NV MEL MEL
NV NV MEL NV MEL MEL NV MEL NV NV
MEL NV NV NV MEL MEL MEL NV NV MEL
MEL NV MEL NV MEL NV MEL NV NV NV

Figure 1: Fifty sample images from the training data.

There are several other ethical problems with deploying such a system into the current healthcare system. First, who is held responsible for a misdiagnosis? Second, how should a doctor's opinion be considered compared to that of the machine? Should the algorithm or the clinician have the last say regarding the diagnosis? Third, how can the algorithm's decision be explained to the patients? Is it enough to say that the algorithm found melanoma, or does anything need to be specified more precisely?

Now, if an algorithm were to go beyond the performance of doctors, how can it be deployed to all, and not only the rich? Who would be in charge of such an algorithm? And if it is trained in one nation and on a majority of its inhabitants, how are the minority groups in those societies affected?

In conclusion, there are several ethical considerations needed for an algorithm predicting diseases such as melanoma. The following topics were mentioned in the report; Evaluation method, sensitivity/specificity, unbalanced or misrepresented training data, classes outside the binary melanoma vs NEV, varying possibilities to get high-quality images, privacy and integrity rights, responsibilities and involvement of computer and human clinician, and the accessibility to a high performing algorithm.

6 Conclusion

The optimal model found was the ConvNeXt. It reached an accuracy of 0.88 on the test data. Though this is relatively high, there are several limitations to the models, most of them are tied to the limited time during which the models were developed and the associated computational effort. The model also has some ethical limitations. For example, concerning which groups of people are represented in the training data. Several other ethical considerations are needed if any such model is to be deployed, for example, who has access to the tool and maintaining patient integrity.

References

Noel Codella, Veronica Rotemberg, Philipp Tschandl, M. Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, Harald Kittler, and Allan Halpern. 2019. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic).

Devron. How to choose the optimal kernel size.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s.

Avery Shepherd. 2021. Convolutional neural network for skin cancer classification - exploring image classification techniques for skin cancer diagnoses using the ham-10000 dataset. *Medium*.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition.

Ali Venosa. 2018. Dangerous melanoma: It's a matter of timing.

Wanshun Wong. 2021. What is residual connection? Accessed: 2024-03-05.

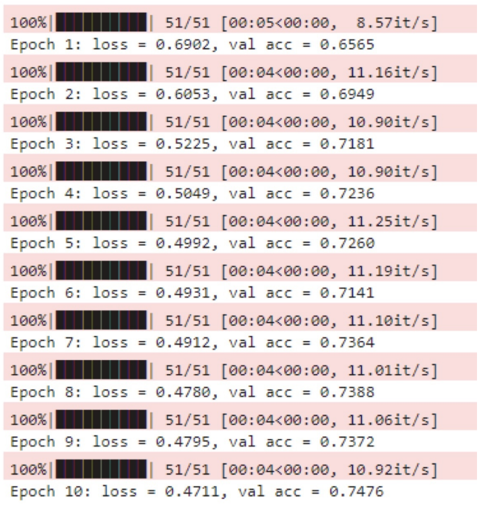
A. Zhang, L. Xing, J. Zou, and et al. 2022. Shifting machine learning for healthcare from development to deployment and from models to data. *Nat. Biomed. Eng*, 6:1330–1345.

A Layers of baseline model

Layer	Parameters
Conv1	In: 3, Out: 64, Kernel: 5
LeakyReLU1	Negative slope: 0.01
MaxPool1	Kernel size: 2
Conv2	In: 64, Out: 32, Kernel: 5
LeakyReLU2	Negative slope: 0.01
MaxPool2	Kernel size: 2
Conv3	In: 32, Out: 16, Kernel: 5
LeakyReLU3	Negative slope: 0.01
MaxPool3	Kernel size: 2
Conv4	In: 16, Out: 8, Kernel: 3
LeakyReLU4	Negative slope: 0.01
MaxPool4	Kernel size: 2
Flatten	-
Linear1	In: 200, Out: 32
LeakyReLU5	Negative slope: 0.01
Linear2	In: 32, Out: 1
Sigmoid	-

Table 2: Layers of the Baseline model.

B Training Baseline



```
100%|██████████| 51/51 [00:05<00:00,  8.57it/s]
Epoch 1: loss = 0.6902, val acc = 0.6565
100%|██████████| 51/51 [00:04<00:00, 11.16it/s]
Epoch 2: loss = 0.6053, val acc = 0.6949
100%|██████████| 51/51 [00:04<00:00, 10.90it/s]
Epoch 3: loss = 0.5225, val acc = 0.7181
100%|██████████| 51/51 [00:04<00:00, 10.90it/s]
Epoch 4: loss = 0.5049, val acc = 0.7236
100%|██████████| 51/51 [00:04<00:00, 11.25it/s]
Epoch 5: loss = 0.4992, val acc = 0.7260
100%|██████████| 51/51 [00:04<00:00, 11.19it/s]
Epoch 6: loss = 0.4931, val acc = 0.7141
100%|██████████| 51/51 [00:04<00:00, 11.10it/s]
Epoch 7: loss = 0.4912, val acc = 0.7364
100%|██████████| 51/51 [00:04<00:00, 11.01it/s]
Epoch 8: loss = 0.4780, val acc = 0.7388
100%|██████████| 51/51 [00:04<00:00, 11.06it/s]
Epoch 9: loss = 0.4795, val acc = 0.7372
100%|██████████| 51/51 [00:04<00:00, 10.92it/s]
Epoch 10: loss = 0.4711, val acc = 0.7476
```

Figure 2: Results of baseline model.

C Layers with Normalization

Layer	Parameters
Conv1	In: 3, Out: 64, Kernel: 5
BatchNorm2d	-
LeakyReLU1	Negative slope: 0.01
MaxPool1	Kernel size: 2
Conv2	In: 64, Out: 32, Kernel: 5
BatchNorm2d	-
LeakyReLU2	Negative slope: 0.01
MaxPool2	Kernel size: 2
Conv3	In: 32, Out: 16, Kernel: 5
BatchNorm2d	-
LeakyReLU3	Negative slope: 0.01
MaxPool3	Kernel size: 2
Conv4	In: 16, Out: 8, Kernel: 3
BatchNorm2d	-
LeakyReLU4	Negative slope: 0.01
MaxPool4	Kernel size: 2
Flatten	-
Linear1	In: 200, Out: 32
LeakyReLU5	Negative slope: 0.01
Linear2	In: 32, Out: 1
Sigmoid	-

Table 3: Layers of the model with batch normalization.

D Layers Residual Connections

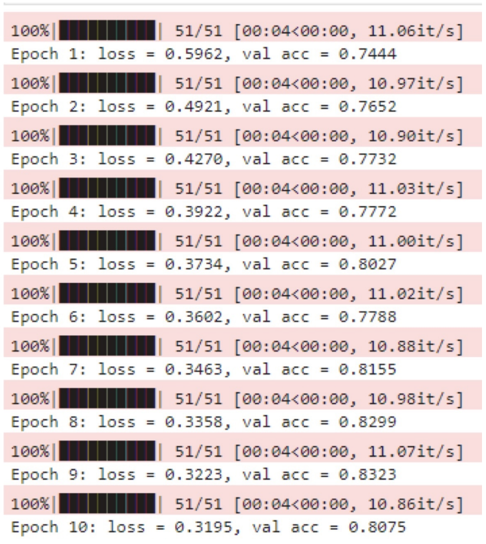
Layer	Parameters
Input	-
1x1 Conv1	In: 3, Out: 64, Kernel: 1
Conv1	In: 64, Out: 64, Kernel: 5, Padding: 2
LeakyReLU1	Negative slope: 0.01
MaxPool1	Kernel size: 3, Stride: 1, Padding: 1
Add (residual1 + x)	-
1x1 Conv2	In: 64, Out: 64, Kernel: 1
Conv2	In: 64, Out: 64, Kernel: 5, Padding: 2
LeakyReLU2	Negative slope: 0.01
MaxPool2	Kernel size: 3, Stride: 1, Padding: 1
Add (residual2 + x)	-
1x1 Conv3	In: 64, Out: 64, Kernel: 1
Conv3	In: 64, Out: 64, Kernel: 5, Padding: 2
LeakyReLU3	Negative slope: 0.01
MaxPool3	Kernel size: 3, Stride: 1, Padding: 1
Add (residual3 + x)	-
1x1 Conv4	In: 64, Out: 64, Kernel: 1
Conv4	In: 64, Out: 64, Kernel: 3, Padding: 1
LeakyReLU4	Negative slope: 0.01
MaxPool4	Kernel size: 1
Add (residual4 + x)	-
Flatten	-
Linear1	In: 1048576, Out: 32
LeakyReLU5	Negative slope: 0.01
Linear2	In: 32, Out: 1
Sigmoid	-

Table 4: Layers residual same input and output size

Layer	Parameters
Input	-
Conv1	In: 3, Out: 64, Kernel: 5, Padding: 2
LeakyReLU1	Negative slope: 0.01
MaxPool1	Kernel size: 3, Stride: 1, Padding: 1
1x1 Conv1(residual1)	In: 3, Out: 64, Kernel: 1
Add (residual1 + x)	-
Conv2	In: 64, Out: 32, Kernel: 5, Padding: 2
LeakyReLU2	Negative slope: 0.01
MaxPool2	Kernel size: 3, Stride: 1, Padding: 1
1x1 Conv2(residual2)	In: 64, Out: 32, Kernel: 1
Add (residual2 + x)	-
Conv3	In: 32, Out: 16, Kernel: 5, Padding: 2
LeakyReLU3	Negative slope: 0.01
MaxPool3	Kernel size: 3, Stride: 1, Padding: 1
1x1 Conv3(residual3)	In: 32, Out: 16, Kernel: 1
Add (residual3 + x)	-
Conv4	In: 16, Out: 8, Kernel: 3, Padding: 1
LeakyReLU4	Negative slope: 0.01
MaxPool4	Kernel size: 1
1x1 Conv4(residual4)	In: 16, Out: 8, Kernel: 1
Add (residual4 + x)	-
Flatten	-
Linear1	In: 131072, Out: 32
LeakyReLU5	Negative slope: 0.01
Linear2	In: 32, Out: 1
Sigmoid	-

Table 5: Layers residual reshaped

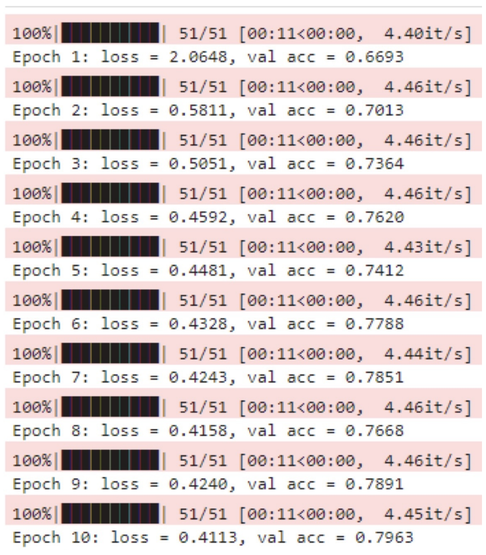
E Training Normalization



```
100%|██████████| 51/51 [00:04<00:00, 11.06it/s]
Epoch 1: loss = 0.5962, val acc = 0.7444
100%|██████████| 51/51 [00:04<00:00, 10.97it/s]
Epoch 2: loss = 0.4921, val acc = 0.7652
100%|██████████| 51/51 [00:04<00:00, 10.90it/s]
Epoch 3: loss = 0.4270, val acc = 0.7732
100%|██████████| 51/51 [00:04<00:00, 11.03it/s]
Epoch 4: loss = 0.3922, val acc = 0.7772
100%|██████████| 51/51 [00:04<00:00, 11.00it/s]
Epoch 5: loss = 0.3734, val acc = 0.8027
100%|██████████| 51/51 [00:04<00:00, 11.02it/s]
Epoch 6: loss = 0.3602, val acc = 0.7788
100%|██████████| 51/51 [00:04<00:00, 10.88it/s]
Epoch 7: loss = 0.3463, val acc = 0.8155
100%|██████████| 51/51 [00:04<00:00, 10.98it/s]
Epoch 8: loss = 0.3358, val acc = 0.8299
100%|██████████| 51/51 [00:04<00:00, 11.07it/s]
Epoch 9: loss = 0.3223, val acc = 0.8323
100%|██████████| 51/51 [00:04<00:00, 10.86it/s]
Epoch 10: loss = 0.3195, val acc = 0.8075
```

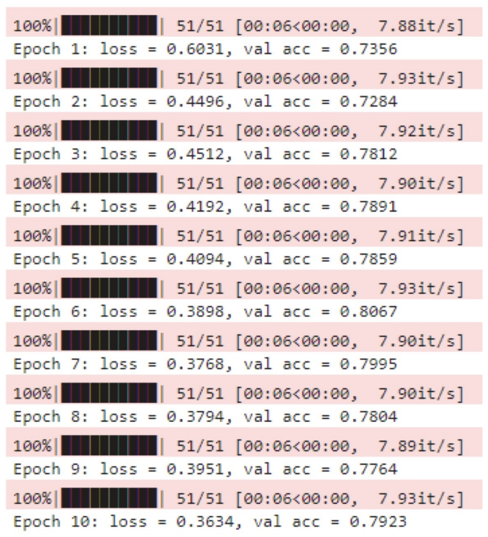
Figure 3: Results of model using normalization.

F Training Residuals



```
100%|██████████| 51/51 [00:11<00:00,  4.40it/s]
Epoch 1: loss = 2.0648, val acc = 0.6693
100%|██████████| 51/51 [00:11<00:00,  4.46it/s]
Epoch 2: loss = 0.5811, val acc = 0.7013
100%|██████████| 51/51 [00:11<00:00,  4.46it/s]
Epoch 3: loss = 0.5051, val acc = 0.7364
100%|██████████| 51/51 [00:11<00:00,  4.46it/s]
Epoch 4: loss = 0.4592, val acc = 0.7620
100%|██████████| 51/51 [00:11<00:00,  4.43it/s]
Epoch 5: loss = 0.4481, val acc = 0.7412
100%|██████████| 51/51 [00:11<00:00,  4.46it/s]
Epoch 6: loss = 0.4328, val acc = 0.7788
100%|██████████| 51/51 [00:11<00:00,  4.44it/s]
Epoch 7: loss = 0.4243, val acc = 0.7851
100%|██████████| 51/51 [00:11<00:00,  4.46it/s]
Epoch 8: loss = 0.4158, val acc = 0.7668
100%|██████████| 51/51 [00:11<00:00,  4.46it/s]
Epoch 9: loss = 0.4240, val acc = 0.7891
100%|██████████| 51/51 [00:11<00:00,  4.45it/s]
Epoch 10: loss = 0.4113, val acc = 0.7963
```

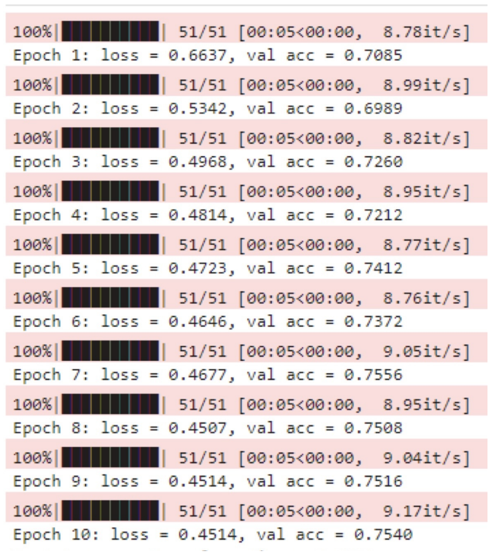
Figure 4: Results of model using residuals with same size.



```
100%|██████████| 51/51 [00:06<00:00,  7.88it/s]
Epoch 1: loss = 0.6031, val acc = 0.7356
100%|██████████| 51/51 [00:06<00:00,  7.93it/s]
Epoch 2: loss = 0.4496, val acc = 0.7284
100%|██████████| 51/51 [00:06<00:00,  7.92it/s]
Epoch 3: loss = 0.4512, val acc = 0.7812
100%|██████████| 51/51 [00:06<00:00,  7.90it/s]
Epoch 4: loss = 0.4192, val acc = 0.7891
100%|██████████| 51/51 [00:06<00:00,  7.91it/s]
Epoch 5: loss = 0.4094, val acc = 0.7859
100%|██████████| 51/51 [00:06<00:00,  7.93it/s]
Epoch 6: loss = 0.3898, val acc = 0.8067
100%|██████████| 51/51 [00:06<00:00,  7.90it/s]
Epoch 7: loss = 0.3768, val acc = 0.7995
100%|██████████| 51/51 [00:06<00:00,  7.90it/s]
Epoch 8: loss = 0.3794, val acc = 0.7804
100%|██████████| 51/51 [00:06<00:00,  7.89it/s]
Epoch 9: loss = 0.3951, val acc = 0.7764
100%|██████████| 51/51 [00:06<00:00,  7.93it/s]
Epoch 10: loss = 0.3634, val acc = 0.7923
```

Figure 5: Results of model using residuals and reshaping.

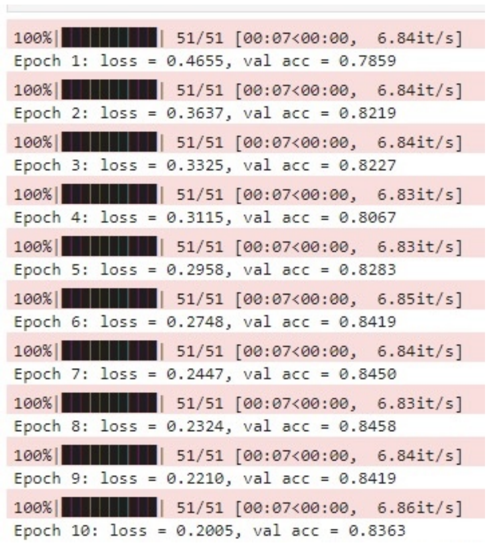
G Training Random Transformations



```
100%|██████████| 51/51 [00:05<00:00,  8.78it/s]
Epoch 1: loss = 0.6637, val acc = 0.7085
100%|██████████| 51/51 [00:05<00:00,  8.99it/s]
Epoch 2: loss = 0.5342, val acc = 0.6989
100%|██████████| 51/51 [00:05<00:00,  8.82it/s]
Epoch 3: loss = 0.4968, val acc = 0.7260
100%|██████████| 51/51 [00:05<00:00,  8.95it/s]
Epoch 4: loss = 0.4814, val acc = 0.7212
100%|██████████| 51/51 [00:05<00:00,  8.77it/s]
Epoch 5: loss = 0.4723, val acc = 0.7412
100%|██████████| 51/51 [00:05<00:00,  8.76it/s]
Epoch 6: loss = 0.4646, val acc = 0.7372
100%|██████████| 51/51 [00:05<00:00,  9.05it/s]
Epoch 7: loss = 0.4677, val acc = 0.7556
100%|██████████| 51/51 [00:05<00:00,  8.95it/s]
Epoch 8: loss = 0.4507, val acc = 0.7508
100%|██████████| 51/51 [00:05<00:00,  9.04it/s]
Epoch 9: loss = 0.4514, val acc = 0.7516
100%|██████████| 51/51 [00:05<00:00,  9.17it/s]
Epoch 10: loss = 0.4514, val acc = 0.7540
```

Figure 6: Results of the model using random transformations on training input.

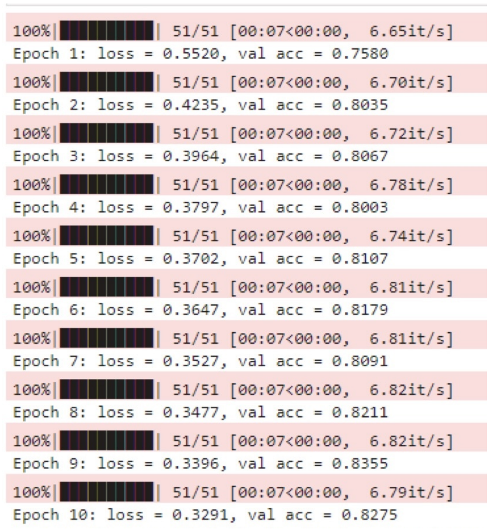
H Training using Residual Blocks and Batch Normalization



```
100%|██████████| 51/51 [00:07<00:00,  6.84it/s]
Epoch 1: loss = 0.4655, val acc = 0.7859
100%|██████████| 51/51 [00:07<00:00,  6.84it/s]
Epoch 2: loss = 0.3637, val acc = 0.8219
100%|██████████| 51/51 [00:07<00:00,  6.84it/s]
Epoch 3: loss = 0.3325, val acc = 0.8227
100%|██████████| 51/51 [00:07<00:00,  6.83it/s]
Epoch 4: loss = 0.3115, val acc = 0.8067
100%|██████████| 51/51 [00:07<00:00,  6.83it/s]
Epoch 5: loss = 0.2958, val acc = 0.8283
100%|██████████| 51/51 [00:07<00:00,  6.85it/s]
Epoch 6: loss = 0.2748, val acc = 0.8419
100%|██████████| 51/51 [00:07<00:00,  6.84it/s]
Epoch 7: loss = 0.2447, val acc = 0.8450
100%|██████████| 51/51 [00:07<00:00,  6.83it/s]
Epoch 8: loss = 0.2324, val acc = 0.8458
100%|██████████| 51/51 [00:07<00:00,  6.84it/s]
Epoch 9: loss = 0.2210, val acc = 0.8419
100%|██████████| 51/51 [00:07<00:00,  6.86it/s]
Epoch 10: loss = 0.2005, val acc = 0.8363
```

Figure 7: Results of model using residual blocks and batch normalization.

I Training using Transformations, Batch Normalization, and Residual Connections



```
100%|██████████| 51/51 [00:07<00:00,  6.65it/s]
Epoch 1: loss = 0.5520, val acc = 0.7580
100%|██████████| 51/51 [00:07<00:00,  6.70it/s]
Epoch 2: loss = 0.4235, val acc = 0.8035
100%|██████████| 51/51 [00:07<00:00,  6.72it/s]
Epoch 3: loss = 0.3964, val acc = 0.8067
100%|██████████| 51/51 [00:07<00:00,  6.78it/s]
Epoch 4: loss = 0.3797, val acc = 0.8003
100%|██████████| 51/51 [00:07<00:00,  6.74it/s]
Epoch 5: loss = 0.3702, val acc = 0.8107
100%|██████████| 51/51 [00:07<00:00,  6.81it/s]
Epoch 6: loss = 0.3647, val acc = 0.8179
100%|██████████| 51/51 [00:07<00:00,  6.81it/s]
Epoch 7: loss = 0.3527, val acc = 0.8091
100%|██████████| 51/51 [00:07<00:00,  6.82it/s]
Epoch 8: loss = 0.3477, val acc = 0.8211
100%|██████████| 51/51 [00:07<00:00,  6.82it/s]
Epoch 9: loss = 0.3396, val acc = 0.8355
100%|██████████| 51/51 [00:07<00:00,  6.79it/s]
Epoch 10: loss = 0.3291, val acc = 0.8275
```

Figure 8: Results of model using transformations, batch normalization, and residual connections.