

Convolutional Neural Networks on Tiny ImageNet

Georgy Mkrtchyan
Skoltech, 2022

Abstract

In this work I investigate how different architectures of convolutional Neural networks performs on the classification task of Tiny-ImageNet 200 dataset. Starting from simple several layer architectures and ending with top performers on the ImageNet challenge (ILSVRC) such as Resnet34 and DenseNet161.

Introduction

Image classification task has been increasing in popularity since 2010 where new breakthrough approaches were introduced, such as AlexNet CNN then ResNet18/34/50/101 etc. Due to high quality performance of the models Image identification has become wide popular across all spheres of life. For example, modern businesses use image classification for different purposes. In agricultural sphere computer vision algorithms are used detection of infected agricultures, governments use computer vision algorithms to keep track of citizens and a lot of different applications of the algorithms are to be stated. The first breakthrough was reached by 8-layer AlexNet with approximately 84% accuracy on the ImageNet dataset in 2012, bringing the industry on another level. However, 3 years later ResNet with 152 layers increased the quality of performance in 5 times showing nearly 3.5% error on the same Dataset. In current analysis we are going to get through the pipeline of different architectures and improvements on the sub-challenge of the ImageNet competition - Tiny-ImageNet-200, which consists of 200 of different classes instead of 1000 introduced in the general competition. The structure of the work is following in the Then in Section 1 we are going to explore the description of the dataset and data processing stage. In Section 2 different approaches and their results are going to be compared ending the stated research.

Section 1

Dataset

Tiny-ImageNet data set consists of 100,000 of training and 10,000 of validation pictures for 200 different classes, such as sport car, lampshade, comic book, etc. Each observation is represented by a 64x64 image.

Data preprocessing

As it is known usage of default images may introduce instability to the model performance due to the fact that a model should correctly classify a cat even if it rotated or some part of the cat is overwhelmed by flash from the camera. For that purpose, are images were subject to different augmentations:

- Random Rotation (keep track of the object from different angles)
- Random Horizontal and Vertical flips
- Central Crop (adjust to the partial object detection)
- Resize (Used after Central Crop to adjust image to initial size – 64x64)
- Gaussian Blur (deal with bad quality parts of the images)
- Normalization

Also, I chose 2 different sets of data augmentation to create artificially increase the size of the training set:

1. Random Horizontal Flip, Central Crop(56x56), Resize, Normalization
2. Random Rotation (10 degrees), Random Horizontal and Vertical flips, Gaussian Blur (kernel size=5, sigma= (0.1,2)), Normalization

As a result, I get a training dataset of 200,000 observations, which should have improved the performance of the models. Now , let us move to the analysis of the model and their results.

Section 2

To begin the analysis, it was chosen to introduce some baseline model such that further analysis had a comparison base. After discovering of the paper of J. Ting “Using Convolutional Neural Network for the Tiny ImageNet Challenge” I used the M1 architecture from the article to be the Baseline model:

Name	Architecture	Test Acc.
M1	IMG→ ConvReLU(F5-16)→ MaxPool→ ConvReLU(F3-16)→ MaxPool→ ConvReLU(F3-32)→ MaxPool → FCRReLU(256) → FC(200)	23.8%

Table 1: The 4 convolutional neural networks and their respective test accuracy. The number of parameters are given for each layer. For example, FC(200) means a fully connected layer with 200 neurons, while ConvReLU(FX-Y) is a conv-ReLU layer Y filters of size X.

, Table 1

Baseline

To begin with I introduced batch size of 256 such that it would fit on the GPU and a Learning rate of 0.001. Due to the computational limits a I run all the models on 20 epoch horizons. Also it is important to state that I chose RAdam as an optimizer algorithm with following parameters:

```
RAdam(baseline.parameters(),lr=0.001,betas=(0.9,0.999),eps=1e-08,weight_decay=0.0001)
```

Where weight decay I used as L2 penalty for overfitting and betas for the moving averages of gradient and its square. Such an algorithm was chosen over SGD to easily control the regularization parameter. (*Betas and weight decay are kept constant over the whole analysis*). Also it is important to mention that all the analyzed model in the research are trained from scratch, and no pretrained parameters were used.

First of all, I decided to run the baseline model only on one set of augmentations and on both sets, i.e., in one case using 100,000 observations and in another 200,000 observations in training set to compare the effect of additional augmentations on images. As a result, I got following results:

- Validation accuracy of Baseline on the 1st augmentation set on validation set is almost reaches 22% after 20 epochs and validation loss as well as training loss has an elbow shape and decreasing trend over the whole horizon.
- While baseline on both augmentation sets shows accuracy on validation set of 29.5%² after 7 epochs and then starts to oscillate around 29% for remaining 13 epochs and increasing the validation loss to the initial value meaning that after 7 epochs a model starts to over fit as training loss continuously decreases all through all epochs.

As we can see introduction of additional augmentation and increase in the number of observations led to a 7.5% increase in validation accuracy, which is a huge improvement in context of such a model. Thus, further I only use dataset with both augmentation sets.

On the next step of the analysis, I decided to decrease the learning rate by a factor of 10 to 0.0001 and compare the performance of the baseline on 2 different rates. As it turned out the resulted accuracy on validation set is much lower for lr=0.0001 and reaches nearly 22.8% after 20 epochs, such a result is approximately equal to one obtained by J. Ting in his paper (23.8%), nevertheless we can for sure state that on this architecture lr=0.001 is much better , as even training accuracy for it after 20 epoch is nearly 55% while for lr=0.0001 is 28% , i.e. twice as lower. So, the hypothesis arises that may be on this task learning rate of 0.001 performs better, but such a statement needs to be proven on other architectures.

¹ J. Ting “Using Convolutional Neural Network for the Tiny ImageNet Challenge”

² Appendix. Fig. 3

Now, it is time to improve the architecture. Thus, again I based my choice on the results of J. Ting and used his best performing model (M4, Table 2) to compare with existing baseline.












M4	 ConvReLU(F5-32)  ConvReLU(F5-32)  MaxPool  ConvReLU(F3-64)  ConvReLU(F3-64)  MaxPool  ConvReLU(F3-64)  ConvReLU(F3-128)  Max-Pool  FCReLU(256)  FC(200)
----	---

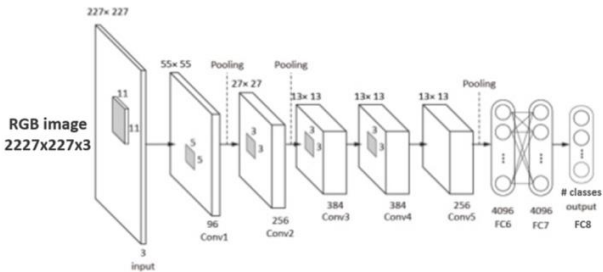
Table 2 ³

Generally, the main difference between Baseline and M4 model is in the number of Convolutional – Activation (ReLU) layers , where first is equipped with 3 while M4 with 6 such layers , while number of max-pooling layers is kept the same, thus the number of estimated parameters is much higher. After estimation of the M4 model with both learning rates following results were obtained:

With learning rate of 0.0001 M4⁴ showed performance on the level of baseline with validation accuracy of 24% and training accuracy of 29%. However, we can see a huge improvement in the quality in comparison to all analyzed models on the M\$ with learning rate of 0.001 it shows validation accuracy of 35.9%, however we can see that validation loss bounces back on th 7th epoch, exactly where validation accuracy starts to oscillate around 35.7%, such a performance shows that a model reaches it's maximum after 7th epoch and then only starts to overfit as training error falls constantly. Generally, the improvement of the quality is described by the increase in the deepness of the architecture.

AlexNet

On the next stage of the analysis, I could not omit the phenomenon called Alexnet, architecture that crushed ImageNet competition in 2012. The architecture consists of 5 Convolution layers of bottleneck form with ReLU activations in-between the layers and initially 3 fully connected layer in the end. However, since the architecture was made for ImageNet competition final fully connected layer is of 1000 dimension thus, I added one more fully connected layer to transform the output from 1000 to 200 dimensions.



⁵Picture 1

As long as we can observe a trend that learning rate of 0.001 is performing better during our analysis, I decided to use it during estimation of Alexnet. Also, I found an observation that after 7 epochs mostly all of the models start to oscillate around some value and validation loss either grows or reaches the

³ J. Ting “Using Convolutional Neural Network for the Tiny ImageNet Challenge”

⁴ Appendix. Fig 5.

⁵ A. Khvostikov, K.Aderghal, J.Benois-Pineau, A.Krylov,G.Catheline - “3D CNN-based classification using sMRI and MD-DTI images for Alzheimer disease studies”,2018

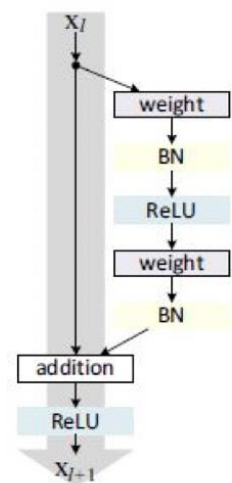
plateau, thus in trials to solve this problem I introduces additional instruments to the analysis – Learning Rate scheduler, which decreases the learning rate by a factor of 10 every 7 epochs. Such an adjustment helps to overcome oscillation, when current gradients stop to improve the performance of the model. The observed results of the AlexNet are not promising as validation error never reached 20% after 20 epochs, moreover, the training error was lower than validation one over the whole horizon. Thus, at that point we move to the ResNet analysis.

ResNet

We are moving towards the most important part of the analysis, which is the implementation of the ResNet CNN. As it was stated above this architecture won the ImageNet Competition in 2015 with error rate of 3.5% on the test data. Of course, due to the computational limits I am not going to use the ResNet152, which consists of 152 layers. However, during the analysis I tried 2 architectures of ResNet: ResNet18 and ResNet34. Both of them are bottleneck shaped consisting of Convolutional layers, max pooling and residual blocks⁶ each consisting of Convolutional layer, Batch normalization and ReLU activation. Such an architecture of the network has an advantage over previously described model, since deep connection of the model allows to capture more feature information a via flow of gradients directly later layers to initial filters. The difference between ResNet18 and ResNet34 is only in the number of convolutional layers and number of residual blocks.

ResNet18

Let us start with ResNet18. To keep up with the logic of the research I estimated the model with two different learning rates $lr_1=0.001$ and $lr_2=0.0001$. For both learning rates I used same scheduler as for AlexNet. ResNet18 with lr_1 currently is showing the best performance among the analyzed model with peak validation accuracy of 45% also outperforming ResNet18 with lr_2 with peak validation accuracy of approximately 34.5% which is equal to performance of M4. The interesting observation is that almost all models reach their peak at the shift of learning rate, i.e. when the learning rate drops by a factor of 10.



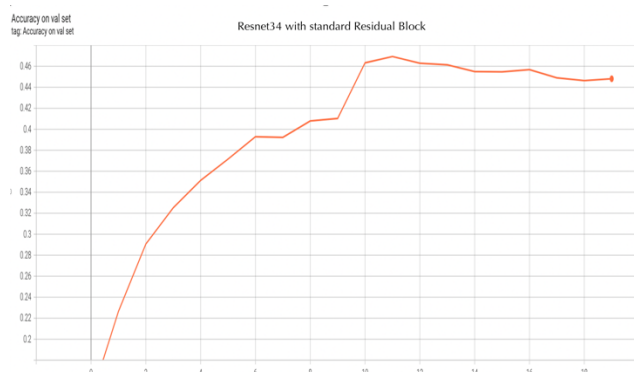
Picture 2 ⁷

ResNet34

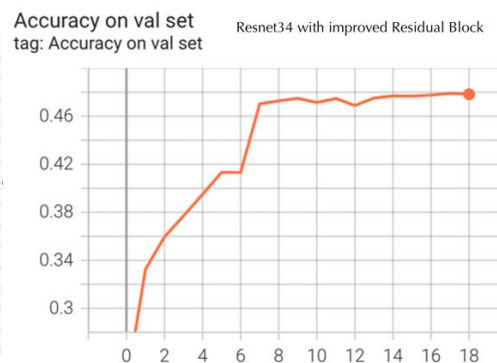
Due to the computational limits, I was able only to run one estimation of the ResNet34, and since learning rate of 0.001 is performing better in our case I decided to use it in this case. However, L. Sun produced an interesting experiment of changing the structure of the residual block from (Picture 2) to full pre-activation structure⁷(Picture 3) where Batch Normalization and activation swap placement with Convolutional layer. According to his finding such a structure of residual block improved the performance of the model. Therefore, I decided to compare the performance of the Resnet34 with lr_2 on 2 different architectures of the residual block. So, the first architecture showed training accuracy of approximately 84% while the validation accuracy since is the best performance model with 46.5% on epoch 12. As we can see on the Graph 2 validation accuracy of Resnet34 with improved residual block shows better performance on the same data with result of 48%.

⁶ Lei Sun – “ ResNet on Tiny ImageNet ”, Figure 9

⁷ Lei Sun – “ ResNet on Tiny ImageNet ”, Figure 9



Graph 1



Graph 2

Moreover, it can be seen the full pre-activation improves the stability of the model as validation error has an increasing trend on the full horizon epochs, which pushes on the thought that further training might have increased the accuracy of the model.

DenseNet161

As a final cherry on the cake, it was decided to use DenseNet with 161 layers. “Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output.”⁸ Such a deep convolutional neural network can stand in one row with such models as ResNet152, as long as DenseNet161 showed 93.8% test accuracy on the ImageNet competition. The time of estimation of one epoch of the DenseNet is 3 times higher than for the previously estimated Resnet34. Thus, estimation process was made only on 8 epochs. However, even with such a number of iterations over the whole dataset a model showed 51% accuracy (Appendix: Figure 12) on the validation set, which is, by far, the best performing model during the analysis.

Conclusion

In this paper we tried to analyze the performance of different architectures of Convolutional Neural Networks on the classification task on the Tiny ImageNet dataset. Firstly, a baseline method was built on the low-quality augmentation training set showed validation accuracy of 22%. After introduction of new images transformations, a target metric has increased by 7.5%, which shows the power of the appropriate data preprocessing. Then different architectures were considered, including ResNet34 with modified residual block, helping the model to increase the validation accuracy from 46% to slightly above 48% percent. Finally, the best result was obtained on the DenseNet161 with 51% validation accuracy after 8 epochs. However, the future investigation are possible since current analysis was limited by the time and space constraints on the usage of the GPU.

⁸ G. Huang, Z. Liu, Kilian Q. Weinberger, Laurens van der Maaten – “Densely Connected Convolutional Networks”, 2018

References

1. G. Huang, Z. Liu, Kilian Q. Weinberger, L. van der Maaten – “Densely Connected Convolutional Networks” 2018
2. Lei Sun –“ ResNet on Tiny ImageNet ”
3. J. Ting “Using Convolutional Neural Network for the Tiny ImageNet Challenge”
4. A. Khvostikov, K.Aderghal, J.Benois-Pineau, A.Krylov,G.Catheline - “3D CNN-based classification using sMRI and MD-DTI images for Alzheimer disease studies”,2018
5. A. Zhai - “Going Deeper on the Tiny Imagenet Challenge”
6. <https://pytorch.org/vision/stable/index.html>
7. **Code References:**
 - a. <https://github.com/pytorch/vision/tree/main/references/classification>
 - b. https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
 - c. <https://www.kaggle.com/code/poonaml/building-resnet34-from-scratch-using-pytorch/notebook> (was used during analysis , but no included in final solution)

Appendix

Fig. 2: Baseline: Augmentation Sets 1, Learning rate=0.001

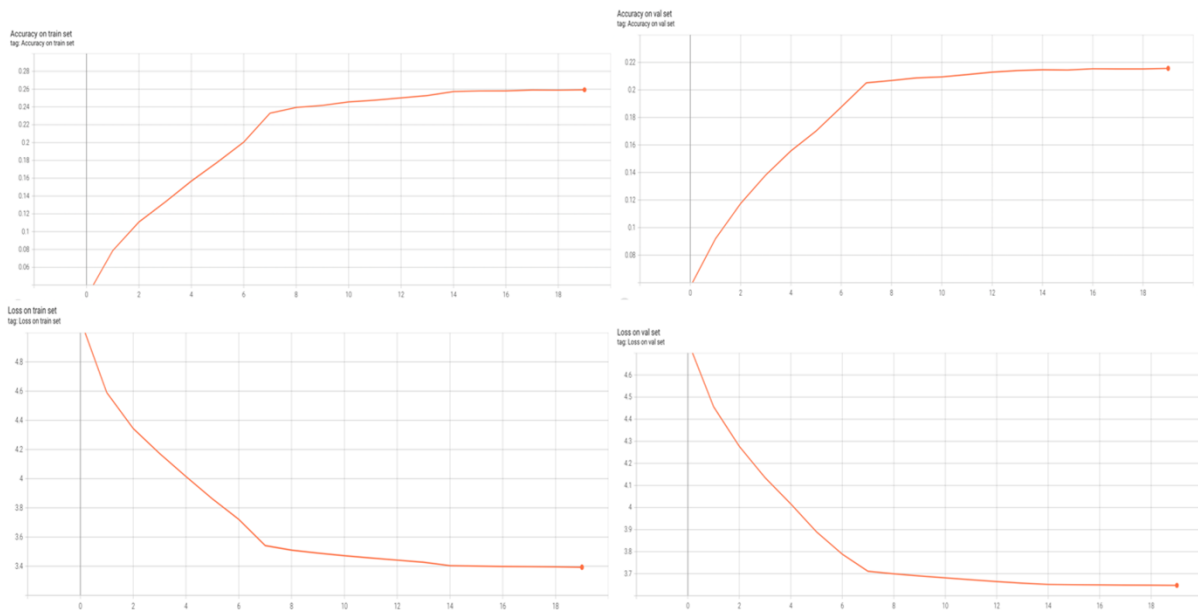


Fig. 3: Baseline: Augmentation Sets 1+2, Learning rate=0.001

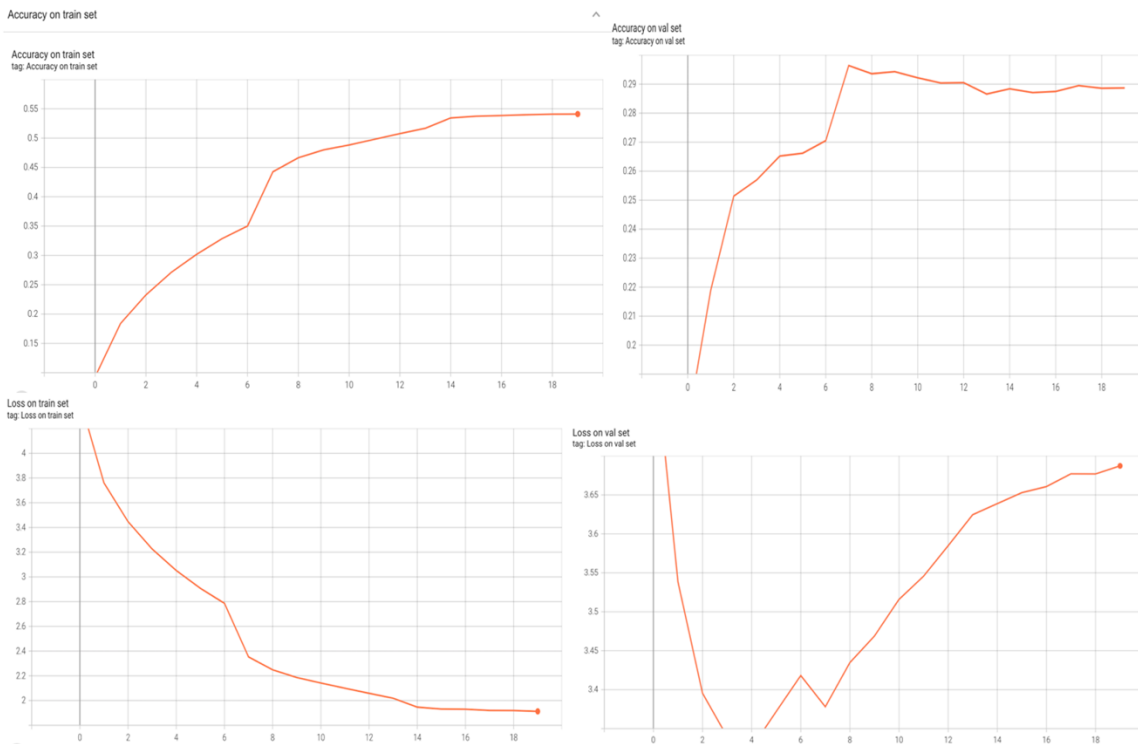


Fig. 4: Baseline: Augmentation Sets 1+2, Learning rate=0.0001

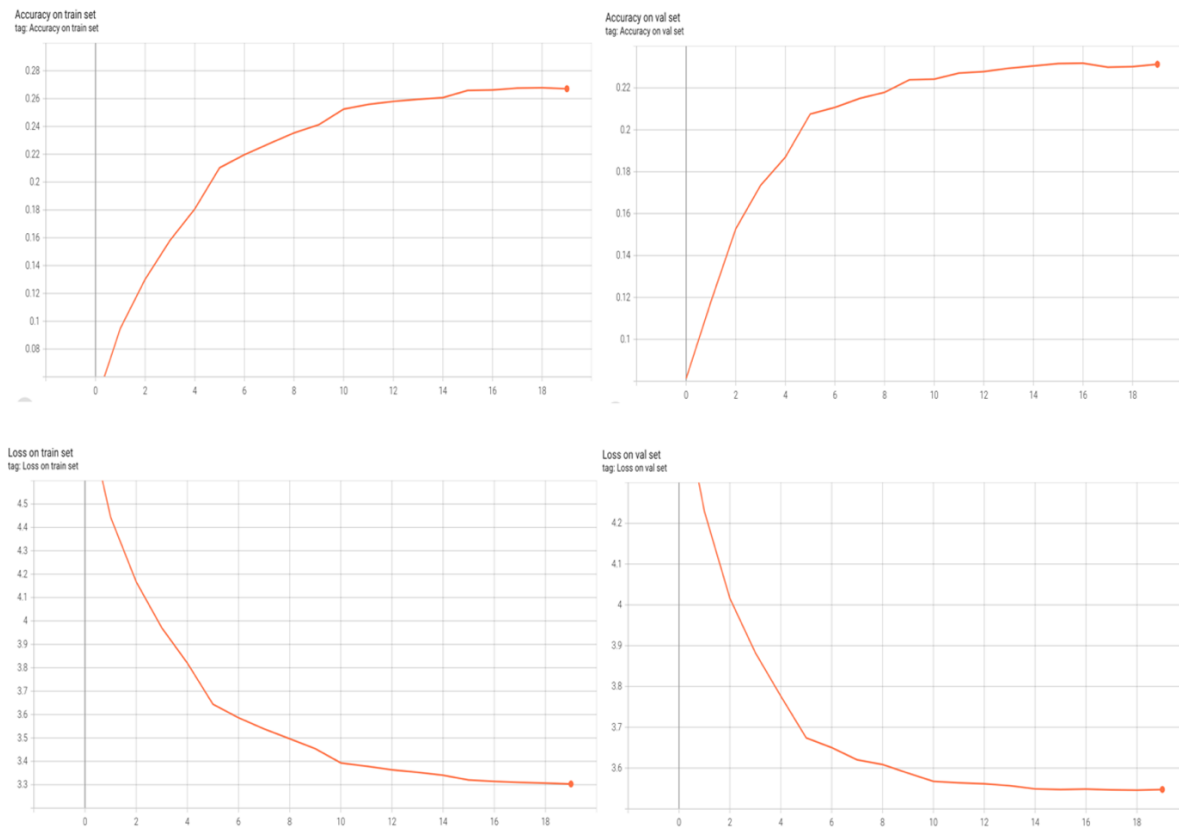


Fig 5. M4: Augmentation Sets 1+2, Learning rate=0.0001

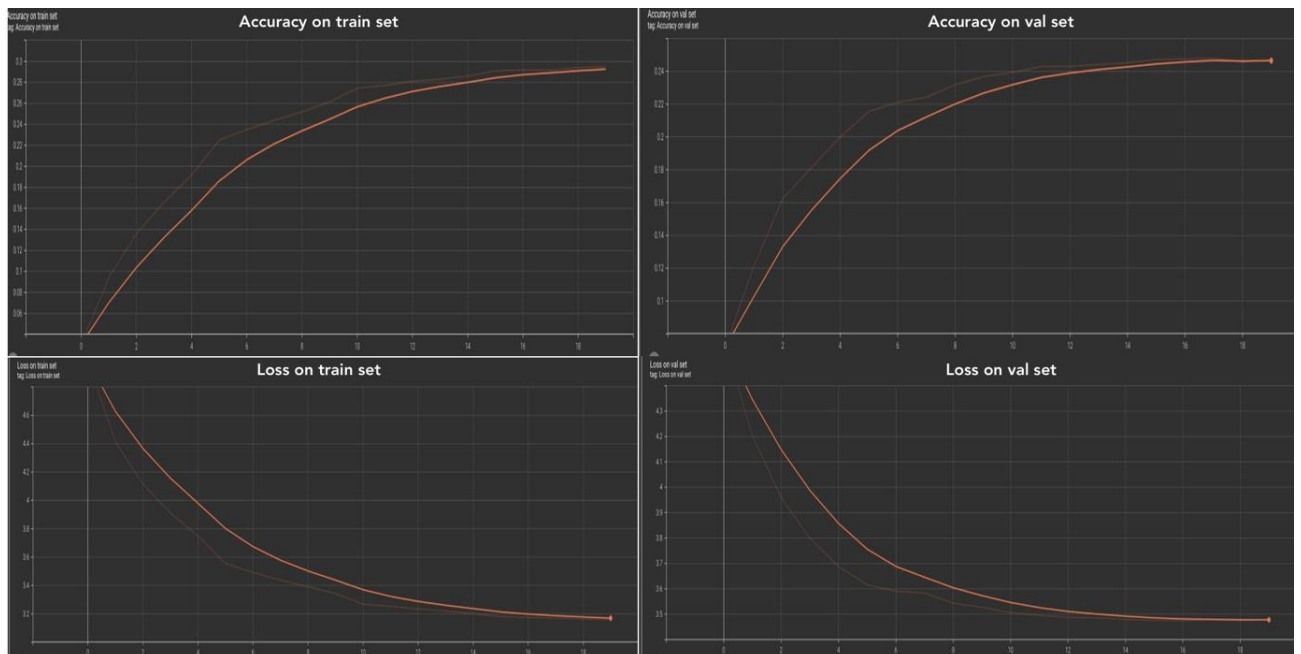


Fig 6. M4: Augmentation Sets 1+2, Learning rate=0.001

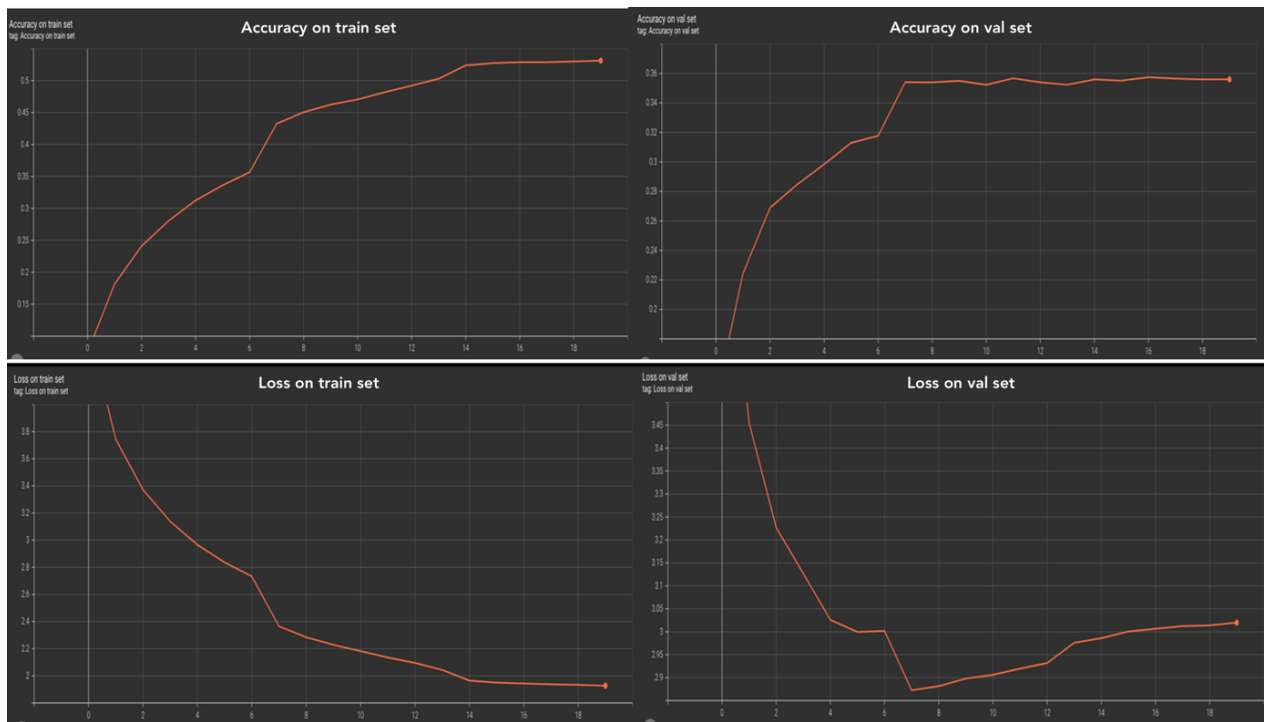


Fig 7. Alexnet: Augmentation Sets 1+2, Learning rate=0.001

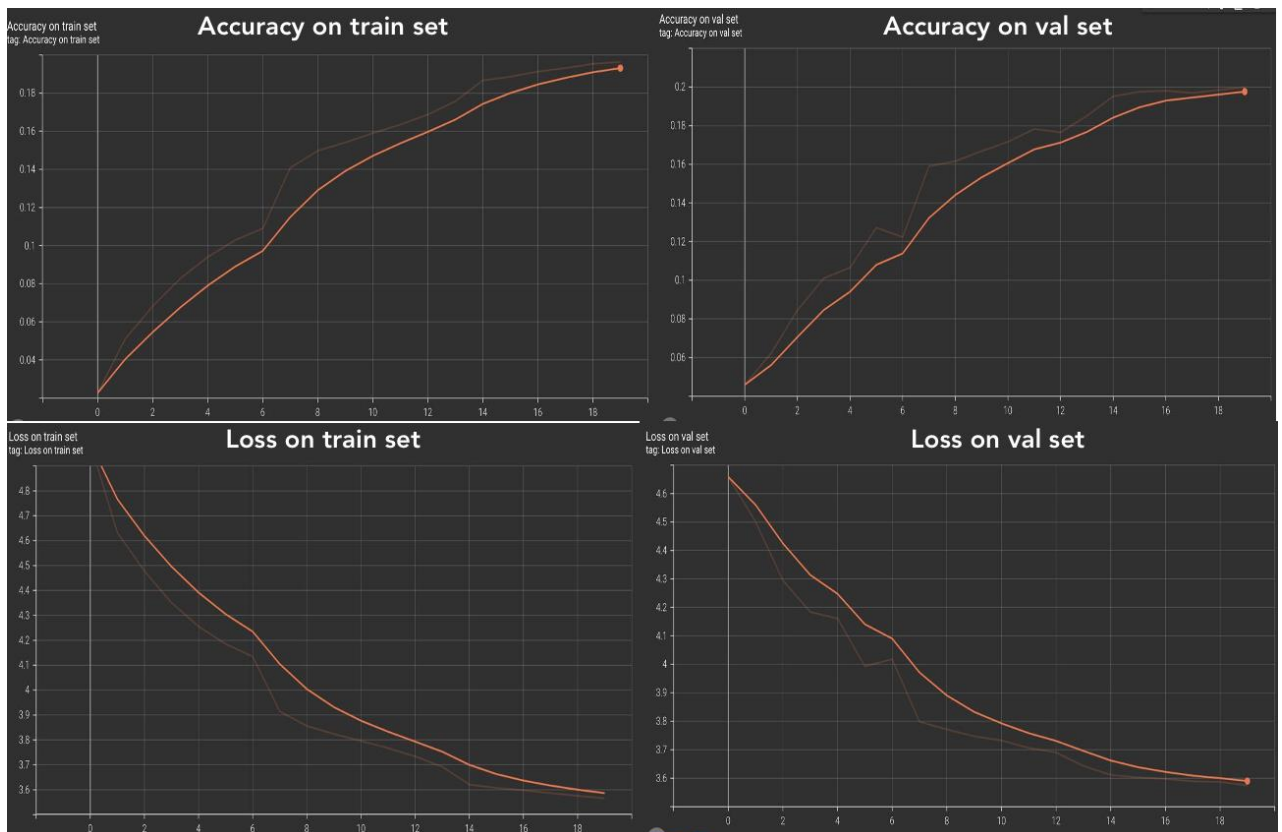


Fig 8. ResNet18: Augmentation Sets 1+2, Learning rate=0.001

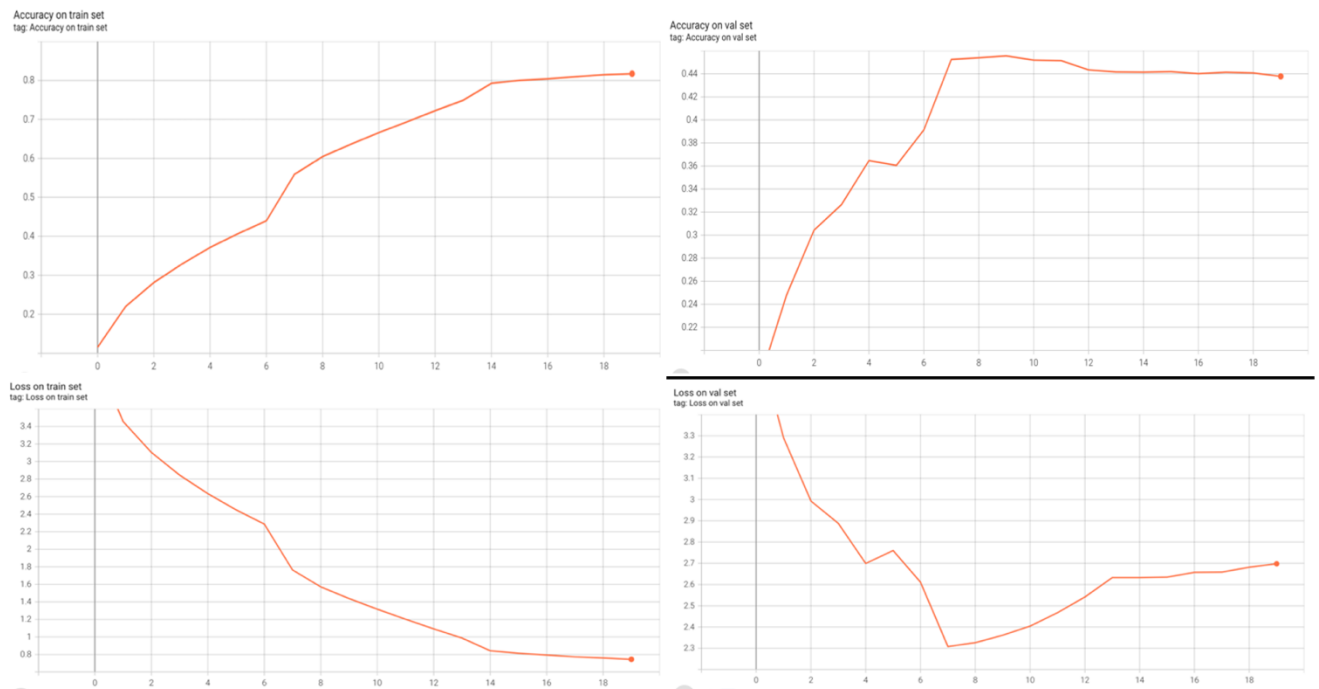


Fig 9. ResNet18: Augmentation Sets 1+2, Learning rate=0.0001

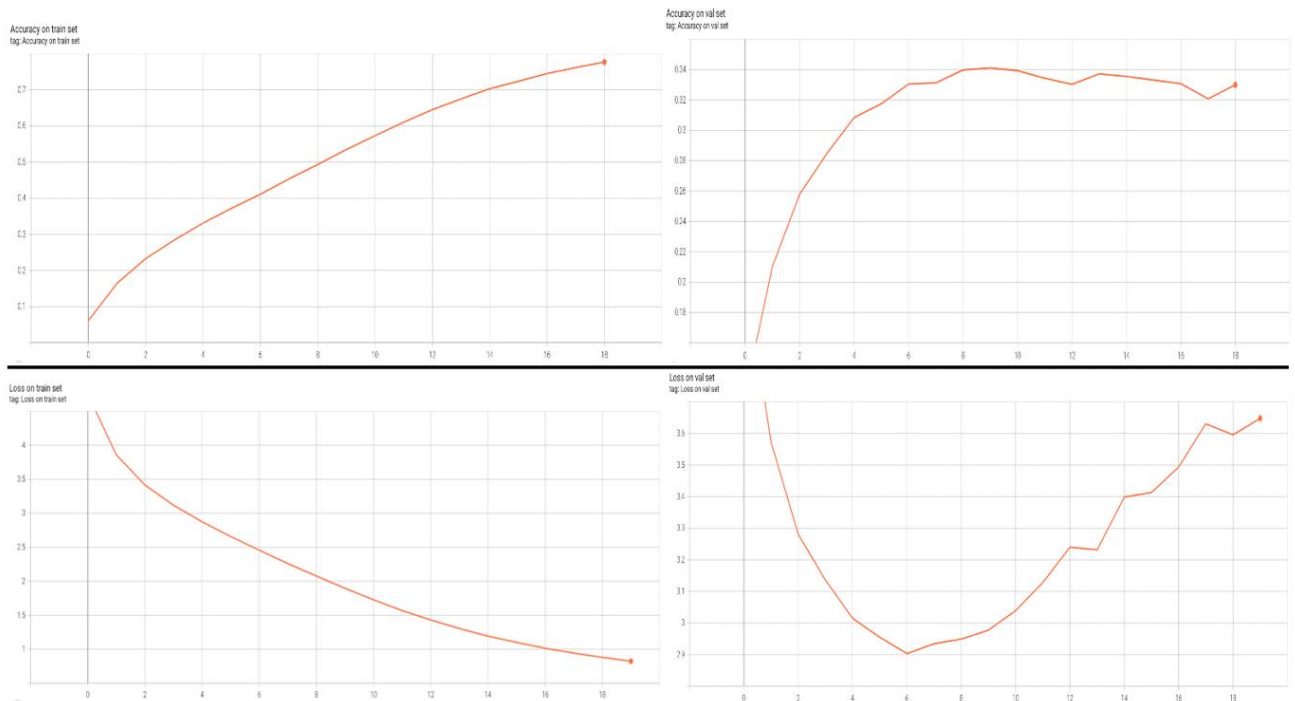


Fig 10. ResNet34 default ResBlock: Augmentation Sets 1+2, Learning rate=0.001

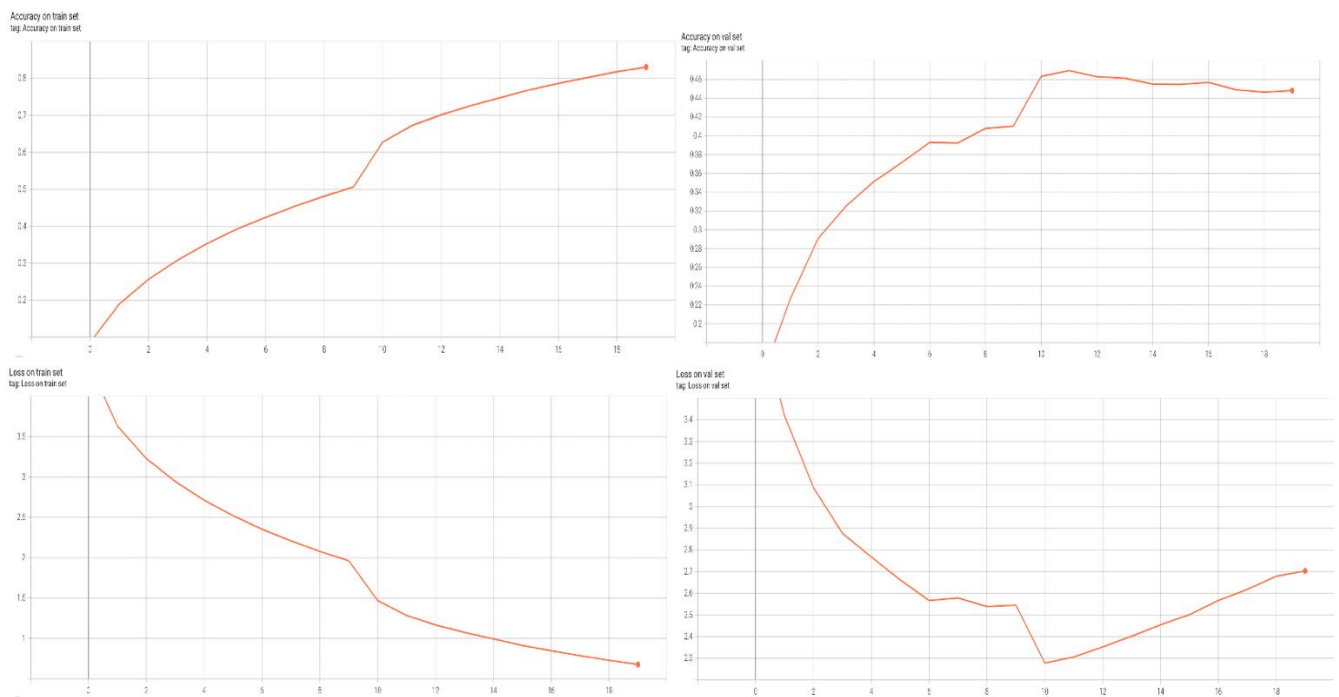


Fig 11. ResNet34 modified ResBlock: Augmentation Sets 1+2, Learning rate=0.001

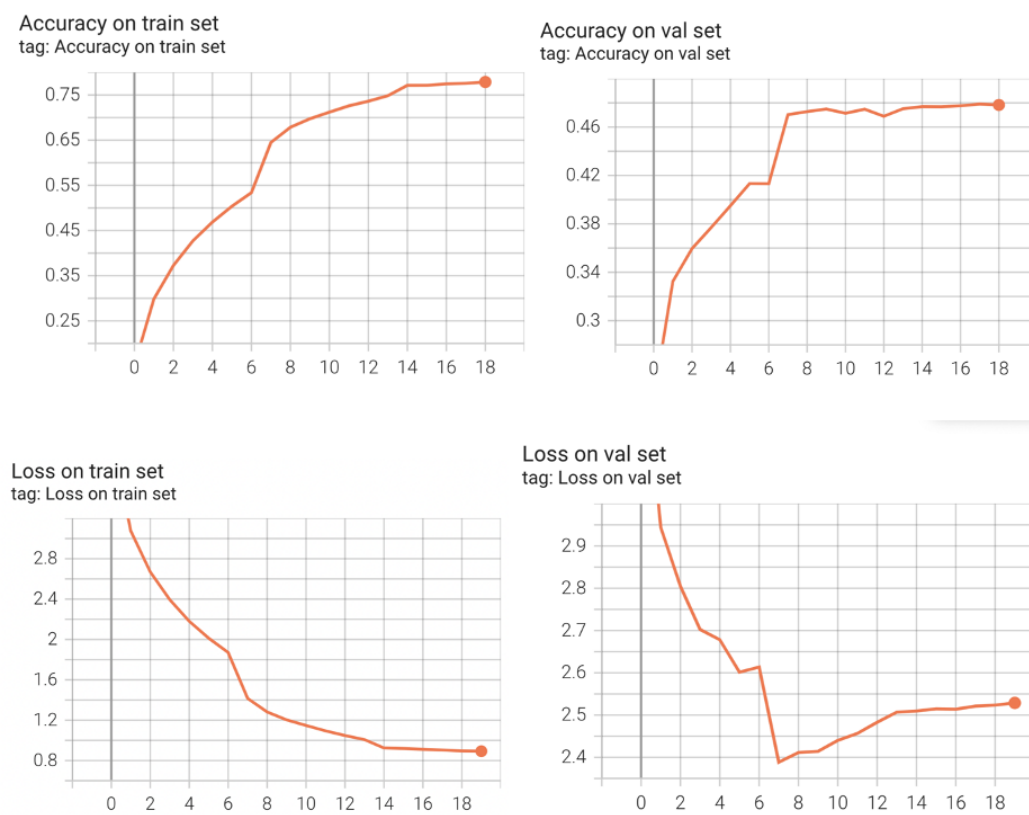


Fig 12. Densenet161: Augmentation Sets 1+2, Learning rate=0.001

