

# Transfer Learning and Spatial Transformer Networks

Amila Ferron

aferron@pdx.edu

## 1 Transfer Learning

Two pretrained models were used, ResNet50 and VGG19, with transfer learning to train categorizing on the MNIST dataset. The weights were frozen for the pretrained models, except for the input weights to the last layer which were replaced with weights going to a layer with 10 nodes and initialized using Uniform Random Distribution from  $\mathcal{U}(-\sqrt{k}, \sqrt{k})$ , where  $k = \frac{1}{\text{in\_features}}$ . These weights were trained on sets with 10, 30, 50, 70, and 90 images of each of the 10 digits. The hyper-parameters are listed in Table 1. These were kept consistent for the two models so may not allow for the optimal accuracy for either. Each model was trained for 50 epochs, with testing run at every epoch. The results are summarized in Table 2, which includes charts of the best accuracy and error for each training set size, and the number of epochs at which the best accuracy and error were achieved for each training set size. The pretrained models were run three times at each training set size to better generalize from the data. Accuracy with the ResNet50 model ranged from 68.56% with 10 training examples of each digit, to 87.80% with 90 examples of each. While accuracy with the VGG19 model ranges from 66.50% with 10 training examples of each, to 88.58% with 90 examples of each. For the accuracy and error, the results consistently indicate that the model pretrained with VGG19 has a quicker early improvement with added training examples although the initial dominance over the ResNet50 model is not maintained, it still achieves greater accuracy than the other model. It would be interesting to increase the training sample size yet again and see if the difference closes entirely. It is worthwhile to note the greater improvement with fewer training samples and in practice, this may be important when sample size is very limited.

The graphs of the number of epochs at which

Hyper-parameter	Value
batch size	4
number of epochs	50
learning rate	0.001
momentum	0.9
learning rate decay step size	7
gamma	0.1
loss function	Cross Entropy
image height	224
image width	224
input channels	3

Table 1: Hyper-parameters for Transfer Learning experiments.

the best accuracy and error were seen are harder to interpret because they are relatively inconsistent. This makes sense because 50 epochs was much more than needed to reach a performance plateau for both models and at any point between roughly 10 and 50 epochs if the noise in the results produces a random jump in performance it can then be recorded as the epoch with the best accuracy. Still, finding a point at which to stop training can be difficult to do. In order to get a more qualitative sense of when the plateau is reached for each model and training set size, graphs of the test results at each epoch were produced. They are shown for the ResNet50 model in Table 3 and for the VGG19 model in Table 4. The graphs for ResNet50 show the plateauing with consistent noise even after many epochs typical of the multi-layer perceptrons and neural networks we have been studying thus far. The VGG19 model in comparison has surprising growth curves, with a lot of noise during improvement and almost no noise once the learning has plateaued. From this, I would think that the epoch at which the model achieves greatest accuracy would be lower and more consistent for

multiple training sessions in the summary graphs in Table 2; and in fact, this is observed in the graphs. Looking at the graph of the number of epochs to get the best error for the VGG19 model, we see a climb with the increase in training sample size that reaches its maximum at a sample size of 50 before declining again. It's hard to know if this trend is present in the ResNet50 model, but the plots of error and accuracy test results per epoch indicate that it's possible there is a similar, if less pronounced trend with that model, as the improvement there appears to stabilize at epochs 8, 14, 11, 10 and 7 for training set sizes of 10, 30, 50, 70, and 90 respectively. If it exists, it appears to be not as significant. The absence of noise in the plateaued VGG19 model results seems like an additional benefit of using this model, because it is a lot easier to see when it has reached its best performance, making it easier to stop training earlier and saving resources. For both models, the results for the first epoch are much better when the training sample size is larger.

After the project requirements were changed, the models were run again for fewer epochs to get plots of the number of epochs to best performance when the model was run for just 15 epochs. The new plots are shown in Figures 1 and 2. With one exception, the models appear to take between 9 and 14 epochs to reach the best accuracy and error rates. This is consistent with the more qualitative findings from viewing the plots of accuracy and error per epoch in Tables 3 and 4.

An additional set of trials was run to tune the hyperparameters. Increasing the batch size, learning rate, and learning rate decay step size was found to improve performance by 1% but also doubled the number of epochs to reach maximum performance. The gains did not seem worth the extra use of resources in this case.

The Colab notebook follows a tutorial ([Chilamkurthy, 2021](#)) and is linked [here](#).

## 2 Spatial Transformer Network

For this section, the model followed the Spatial Transformer Network used in ([Jaderberg et al., 2015](#))

### 2.1 Running the Spatial Transformer Network on Unaltered MNIST Data

Training the Spatial Transformer Network on the MNIST dataset gave an accuracy of 98.89% and

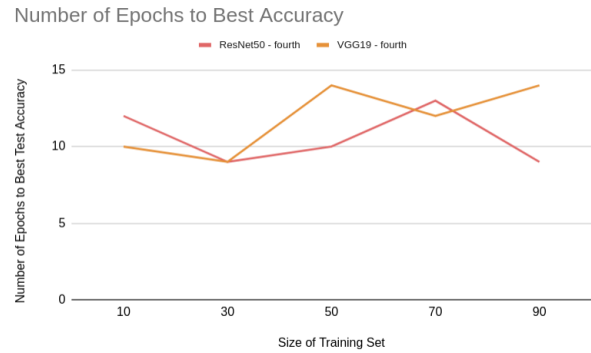


Figure 1: Number of epochs to best accuracy for the ResNet50 and VGG19 models.

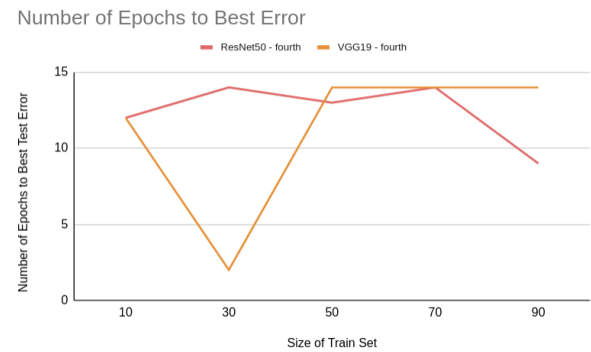


Figure 2: Number of epochs to best loss for the ResNet50 and VGG19 models.

98.63% after 20 epochs over two training and testing trials. Accuracy was high from the first epoch where it achieved 93.09% during the second training and testing trial. Initial accuracy was not recorded for the first trial. These results are impressive and indicate that although the task of finding a useful transformation for each image is complex, it can be accomplished by the model in a short time and give good results. Plots of accuracy and error performed per epoch on the test set is shown in Figures 3 and 4.

An exploration was done to determine if any transformations were used on more than one image, and it was found that each deformation is unique to the image being processed. In order to visualize the transformations being performed, they were printed for a batch of 64 images before and after applying the grid to the features. The types of deformations possible with the method used in the network are in the set of affine transformations. This includes no change (multiplying by the identity matrix); translation (just moving); scaling across the x-axis, y-axis, or both; rotating around the origin; shearing in the x-direction, y-direction, or both; and reflecting over

Table 2: Graphs Summarizing Results for Part 1

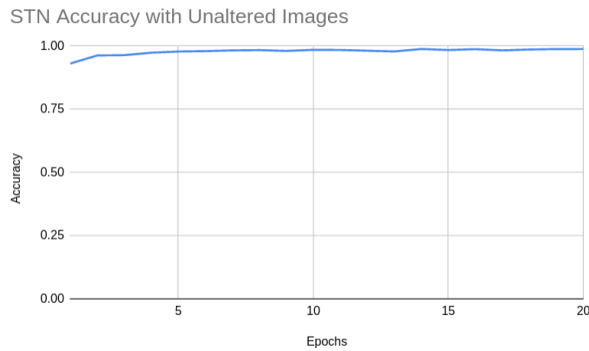
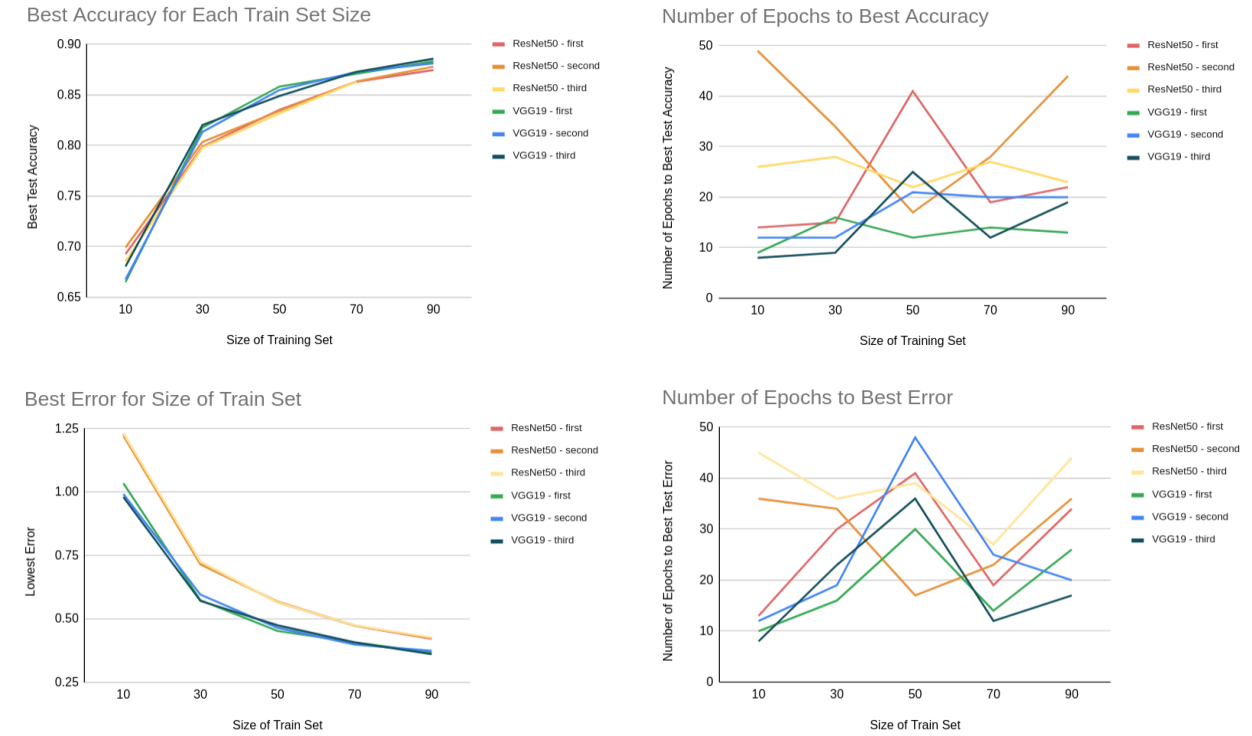


Figure 3: Accuracy per epoch for the STN model used with just unaltered data.

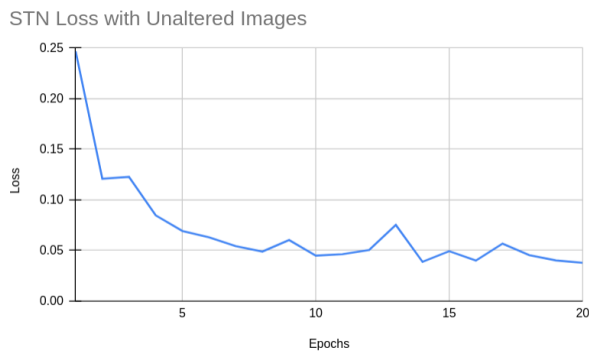


Figure 4: Loss per epoch for the STN model used with just unaltered MNIST data.

the x-axis, y-axis, or both. Other network models can use projection or thin plate spline transformations in addition to affine transformations, but this project was limited to affine transformations.

Examples of these transformations are shown in Table 5. Example 'a' shows shear, but also translation, as can be seen by the uneven spacing on the sides of the image. There is more of the lighter gray on the right side of the altered image than on the left side and more at the bottom than the top. Example 'b' shows shear as well, but the letter also appears to get wider, suggesting that it was scaled in the x-direction. Example 'c' appears to get taller, indicating scaling in the y-direction. Example 'd' looks slightly larger in both directions. Example 'e' shows possible rotation of the image, although this could have also been accomplished with shearing. Example 'f' shows shearing in the x-direction, making the number stand up straighter. Example 'g' shows shear in the y-direction, lifting the flat part of the 7 to be straighter. Example 'h' shows a combination of shear, straightening out the number, and scaling, making it look bigger.

Reflection was not seen in the transformation of the images, likely because the MNIST numbers appear to be all "unreflected". I haven't seen any examples that are the mirror image of the usual way

the numbers are written.

## 2.2 Use of the Spatial Transformer Network on Deformed MNIST Images

The MNIST test and training sets were supplemented with the same images modified with the elastic deform library to perform random elastic deformation and zooming on the image set. These new image sets were then added to the original MNIST data sets to give final sets that are three times the size of the original test and training sets. Examples of these alterations are shown in Table 6. Training was run on the new, larger training data set and tested at each epoch to plot improvement over time. Training and test results are shown in Figures 5 and 6.

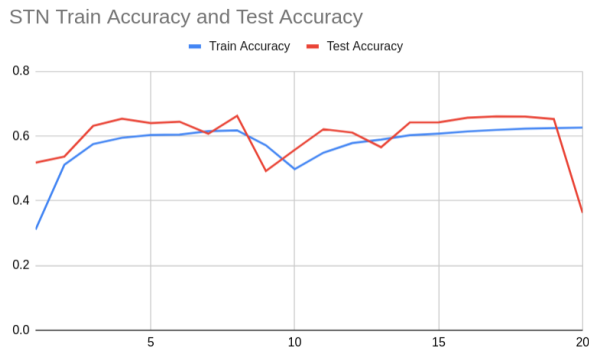


Figure 5: Training and testing accuracy per epoch for the STN model used with altered and unaltered data.

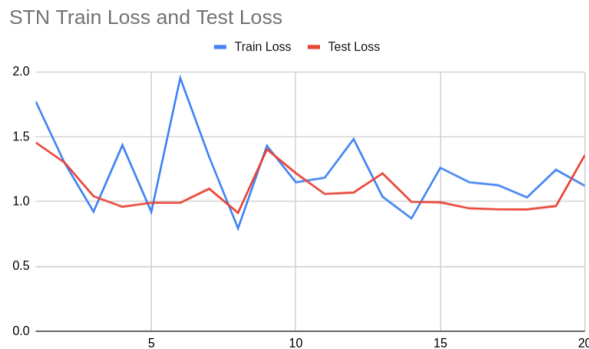


Figure 6: Training and testing loss per epoch for the STN model used with altered and unaltered MNIST data.

The plots for accuracy and loss for just the test sets are shown in the top part of Table 7. The training and testing were run three times to better show trends in the results. In comparison with the 98.89% achieved when training with just the unaltered MNIST images, these results are not as impressive, with a maximum accuracy of 67.96% achieved during the second run of the training and

testing. The tests were also run on separate test sets for each of the deformation types: unaltered, elastic deformed, and zoomed. The results for each deformation type were plotted over two trials and are shown in the bottom portion of Table 7. Testing on unaltered data showed results similar to the results for the test set from Section 2.1, with high values of 98.17% and 98.26%. The results for elastic deformed images were just slightly above random selection with highs of 16.52% 13.40%. The model also performed well on zoomed images with accuracy high values of 89.99% and 90.84%

In order to understand the transformation types that are being applied to the elastic deformed and zoomed images, the images were plotted before and after transformation. The transformations applied to the elastic deformed images do not appear to show digits more clearly, however the transformations applied to zoomed images do make digit identification easier to the human eye. Examples of the transformations are shown in Table 8.

Given the differing results for the deformation types and the appearance of the effect of the transformations on the unaltered, elastic deformed, and zoomed images, it appears that the transformations are effective for the unaltered images and the zoomed images, but not the elastic deformed images. This aligns with the nature of the transformations and that they can only be affine, the way this model is implemented. Because the elastic deformation uses random displacement of the pixels, this type of deformation cannot be undone by an affine transformation, and this is seen in the test results and example images of the transformations. It would be interesting to try the elastic deformed samples with a model that uses thin plate spline transformation and see how it performs.

The Colab notebook for this section follows the tutorial at ([Hamrouni, 2021](#)) and is [here](#).

## References

- Sasank Chilamkurthy. 2021. [Transfer learning for computer vision tutorial](#).
- Ghassen Hamrouni. 2021. [Spatial Transformer Networks Tutorial](#).
- Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. 2015. [Spatial transformer networks](#). *CoRR*, abs/1506.02025.

Table 3: Plots of error and accuracy at each training set size recorded for ResNet50. The improvement appears to stabilize at epochs 8, 14, 11, 10, and 7 at training set sizes of 10, 30, 50, 70, and 90 respectively. Loss is calculated with Cross Entropy Loss and divided by 2.3 to allow it to be plotted in the same range as accuracy.

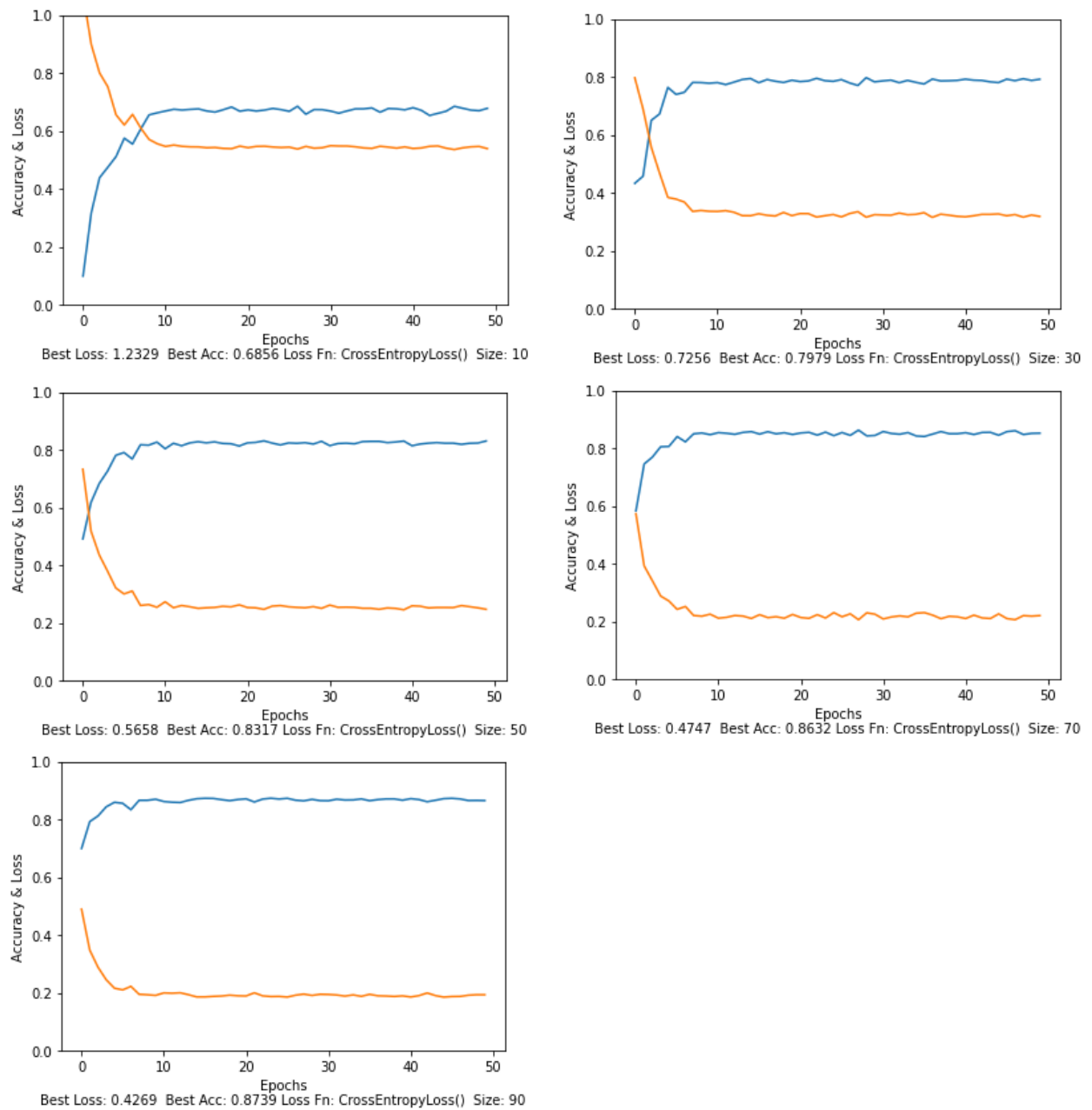
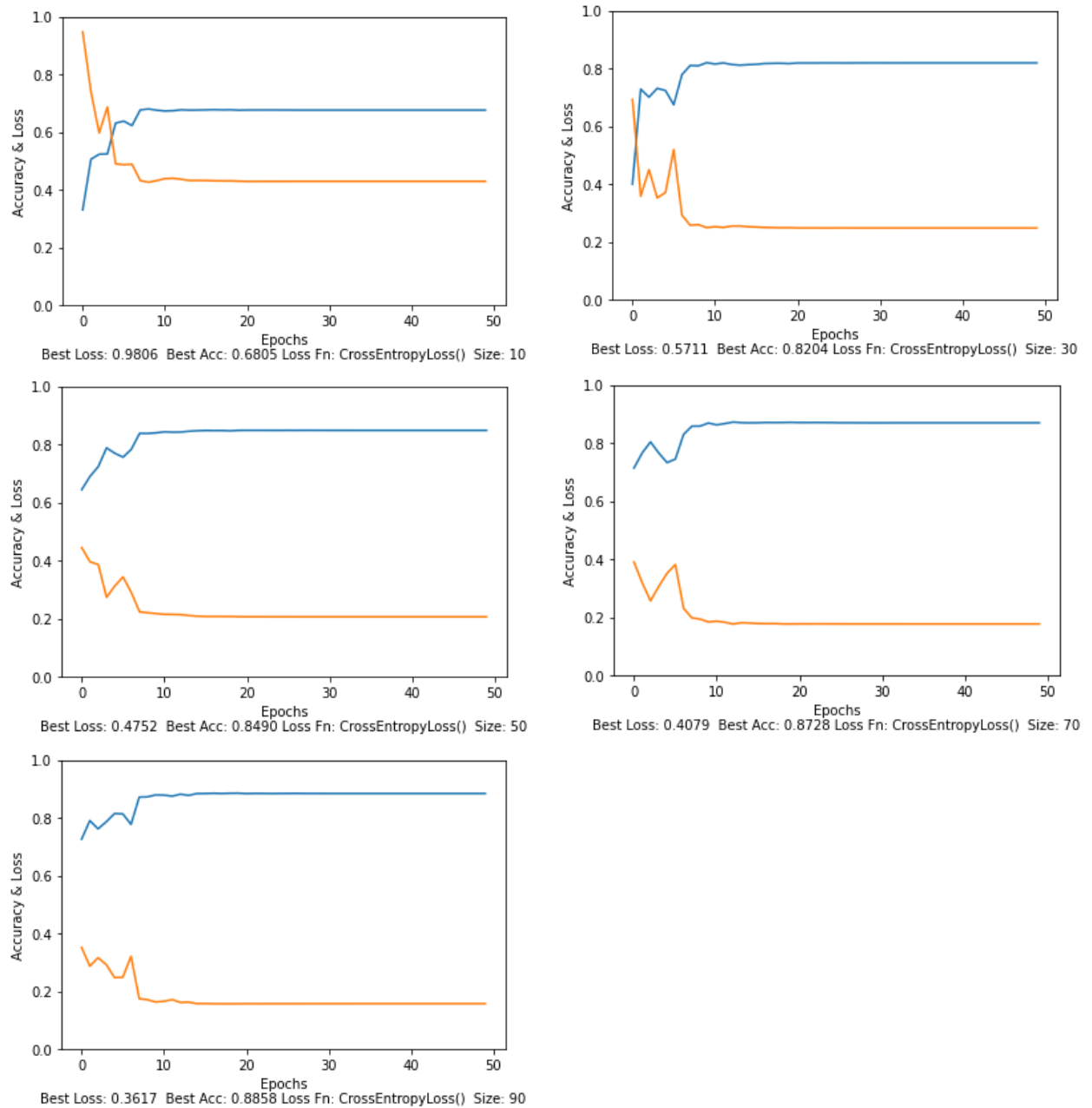


Table 4: Plots of error and accuracy at each training set size recorded for VGG19. The improvement appears to stabilize at epochs 7, 10, 15, 12, and 13 at training set sizes of 10, 30, 50, 70, and 90 respectively. Loss is calculated with Cross Entropy Loss and divided by 2.3 to allow it to be plotted in the same range as accuracy.



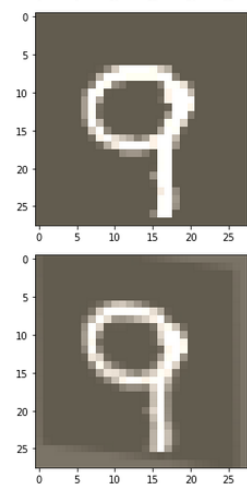
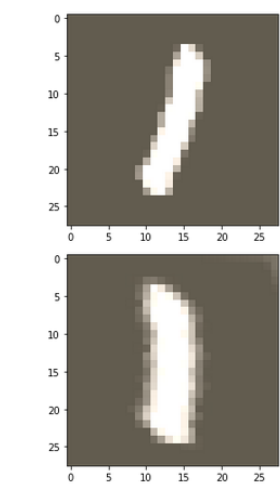
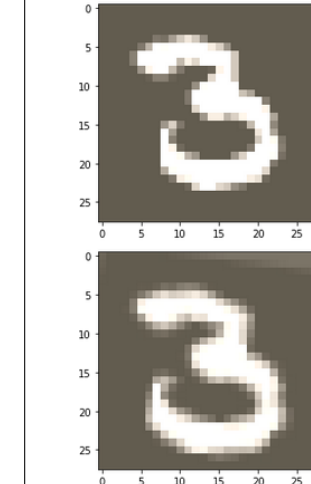
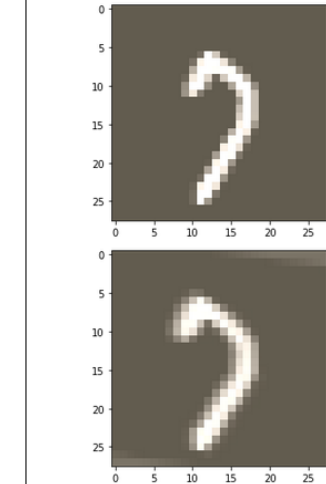
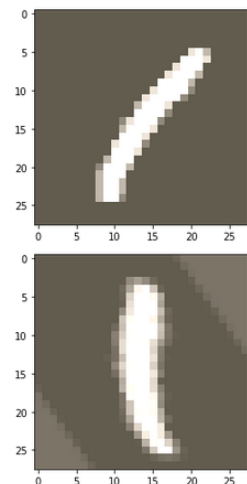
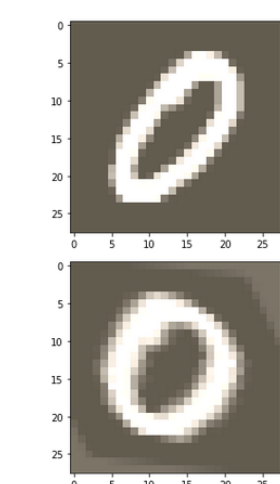
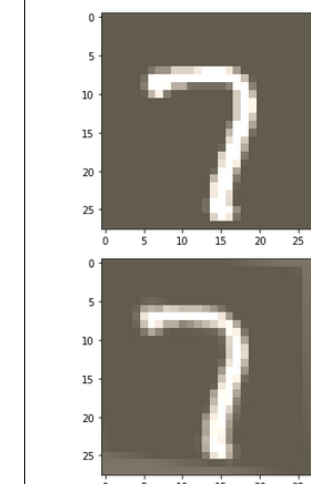
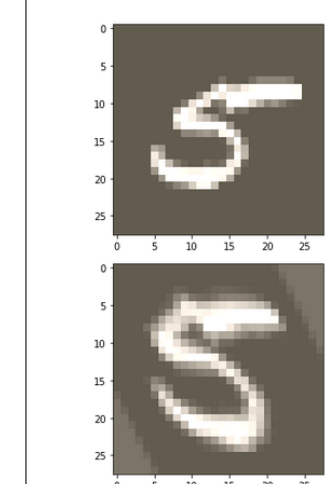
 <p>a. translation</p>	 <p>b. scaling in the x-direction</p>	 <p>c. scaling in the y-direction</p>	 <p>d. scaling both x- and y-axes</p>
 <p>e. rotation</p>	 <p>f. shear in the x-direction</p>	 <p>g. shear in the y-direction</p>	 <p>h. shear and scaling</p>

Table 5: Examples of the types of transformations used by the Spatial Transformer Network.

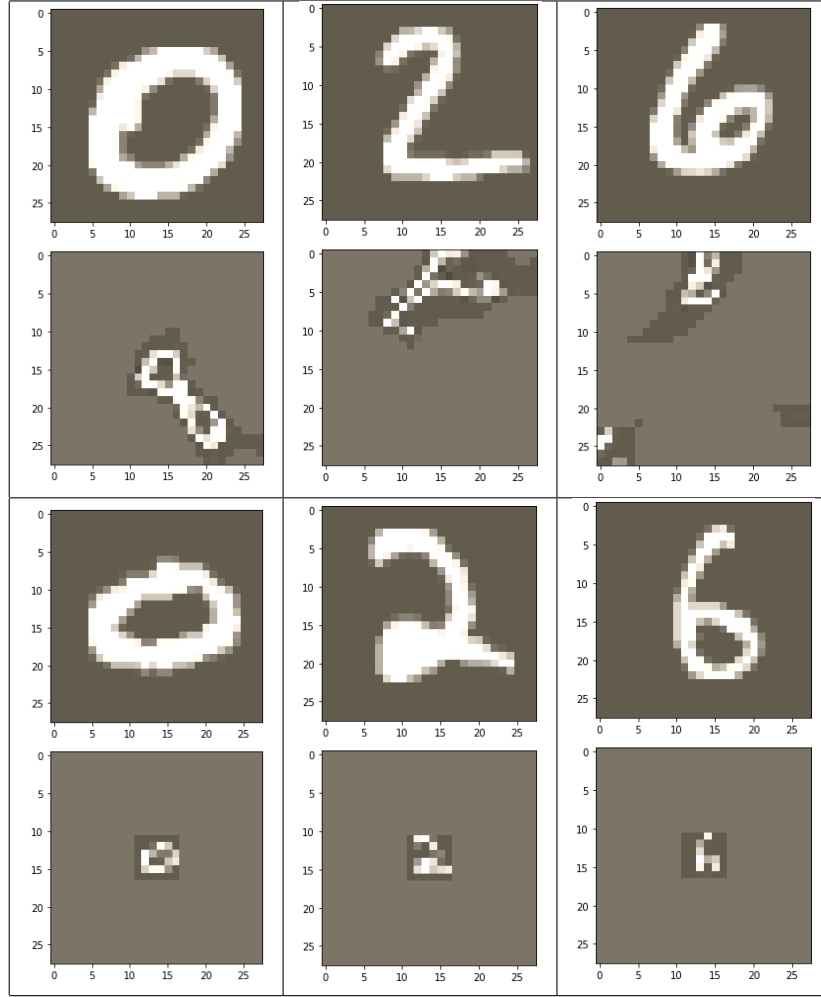


Table 6: Examples of elastic deformations (top row) and zooming out 4x (bottom row). The new images produced were appended to the original data sets.



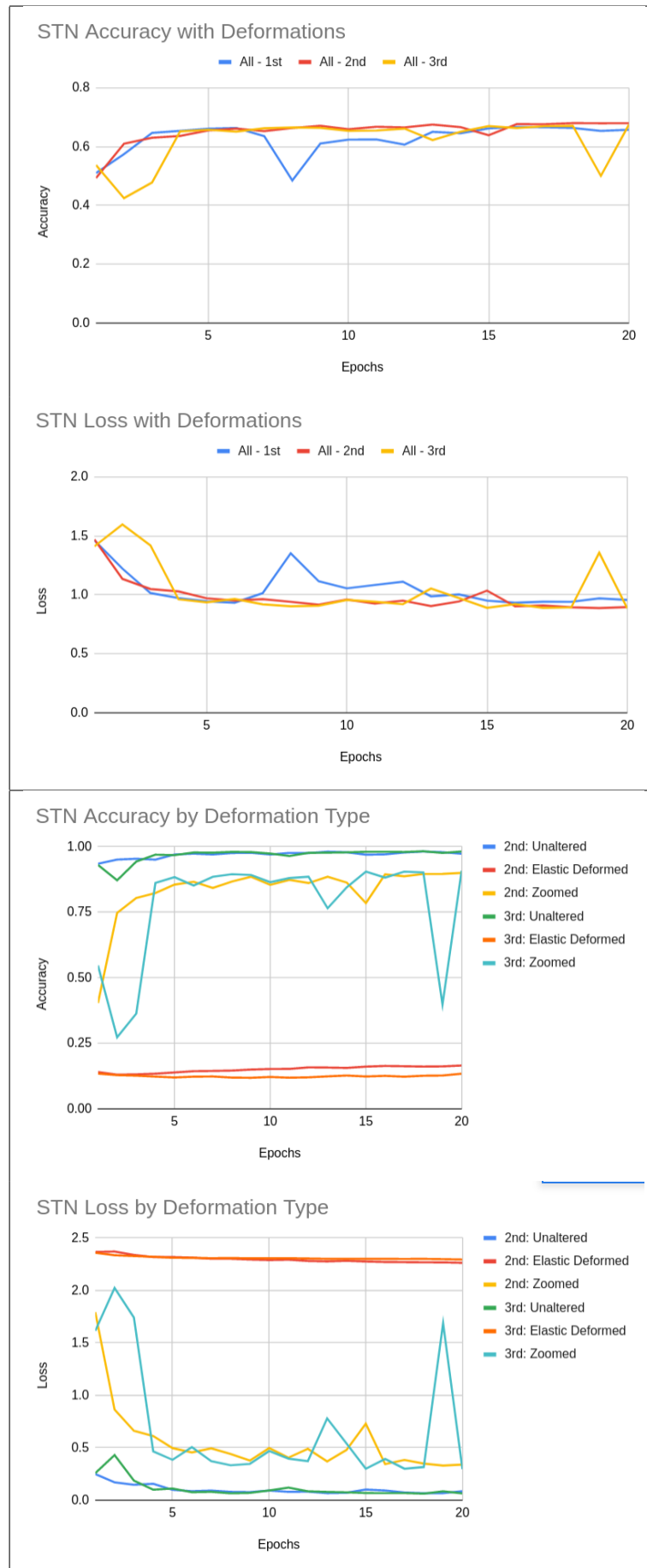


Table 7: Top: The results during training using the augmented data set. Bottom: The results by deformation type for the model trained with the augmented data set.

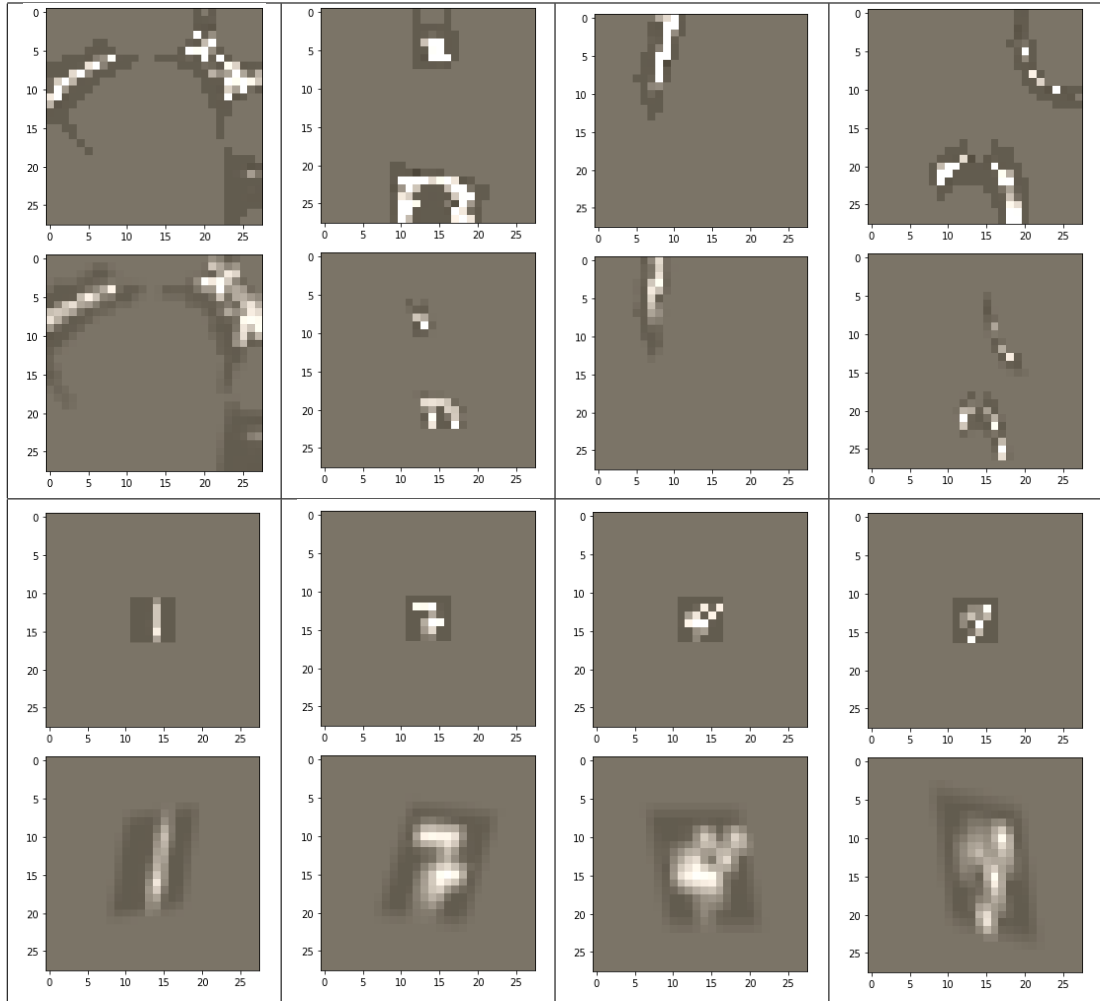


Table 8: Examples of transformations used by the Spatial Transformer Network to classify images modified with elastic deformation (top row) and zooming out 4x (bottom row).