Modified MNIST Handwritten Digit Recognition

Andrei Romascanu
Department of Electrical and Computer Engineering
McGill Universiy
Montreal, QC
andrei.romascanu@mail.mcgill.ca

Hamza Rizwan
Department of Electrical and Computer Engineering
McGill University
Montreal, QC
hamza.rizwan@mail.mcgill.ca

Ashray Malleshachari
Department of Computer Science
McGill University
Montreal, QC
ashray.malleshachari@mail.mcgill.ca

Abstract

The MNIST handwritten digit classification problem is a classic problem used in computer vision and deep learning. The objective of the project was to solve a classification problem of identifying the largest number (by numeric value) in each image in a dataset of images with 3 randomly transformed digits overlaid on noisy backgrounds. To achieve this goal, we ran several experiments testing different model architectures, model sizes, the effect of data augmentation, and also tested performance using pretrained models. We found that ResNet architectures had the best performance, and increased with model size, data augmentation, image upscaling, and model pretraining. Our final test-set accuracy was 97.7%.

1 Introduction

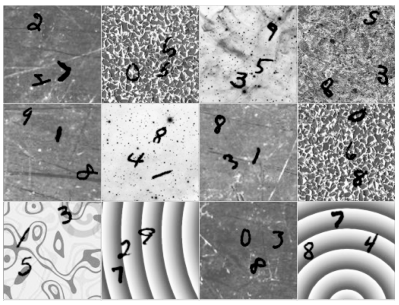


Figure 1: Example images from the dataset.

The modified MNIST task involves predicting the numerically largest digit in image composed of randomly transformed digits overlaid on noisy backgrounds. Figure 1 shows example images from the our modified dataset. Our goal was to create a supervised classification model for this dataset and optimize performance on a withheld test set. To achieve this, we leveraged modern deep learning computer vision techniques and ran controlled experiments to validate design decisions on model architecture and size, data augmentation and image resizing, and model pretraining. We kept other design decisions constant, such as using Adam as an optimizer. We found that the ResNet architectures performed best out of all the architectures we tested; the model size increased performance, but that there were diminishing returns from increased model depth which could be side-stepped with increased width; we also found that data augmentation by random image transforms during training had a pronounced regularizing effect and significantly improved validation performance; and finally, model pretraining on ImageNet improved validation performance. In the end, our final submission obtained a test set accuracy of 97.7%.

2 Related Work

The MNIST dataset has gained a lot of prominence in the field of image classification. For our project, we have referenced some well known models such as ResNet, MobileNet and ShuffleNet, along with a simple CNN as our benchmark.

Convolutional Neural Networks (CNNs) were inspired by biological processes and are based on research done by Hubel et al [5] regarding the visual system's structure in cats and monkeys. This further inspired K Fukushima to develop a hierarchical, multilayered artificial CNN in 1980 called Neocognitron[6]. Further progress can be seen in the work of LeCun et al [7][8] which incorporates CNNs with the error gradient, achieving very good results in a variety of pattern recognition tasks.

Deep convolutional neural networks have led to a series of breakthroughs for image classification, in some cases right up to or very close to human competitive performance. This has resulted in the proposal of various new models, as well as the creation of benchmark datasets. But a degradation problem [2] has been observed when deeper networks are able to start converging; as the network depth increases, accuracy gets saturated and then degrades rapidly.

ResNet, proposed by He et al in 2015 [4],revolutionized the CNN architectural race by introducing the concept of residual learning in CNN and devised an efficient methodology where the layers of a deep neural network were rearranged as residual functions with a reference to layer inputs with a much simpler structure. This 152 layer model was tested on the ImageNet Dataset and won first place at the ILSVRC-2015 and COCO-2015 competition, achieving an error of 3.75%.Thus it can reduce the training error while also deepening network depth, and also solves the problem of gradient dispersion [3], all while improving the overall performance.

The above models, while improving accuracy did not necessarily make networks more efficient with respect to computational size and speed, particularly for many real world applications. To deal with these issues,Howard et al [9] proposed an efficient network architecture and a set of two hyper-parameters to build very small, low latency models that can cater to the design requirements

for mobile and embedded vision applications called MobileNets.They have two novel features: usage of depth-wise separable convolutions to build light weight deep neural networks and addition of two simple global hyper-parameters i.e. the width multiplier and the resolution multiplier that efficiently trade off between latency and accuracy. These hyper-parameters let the model builder to choose the right sized model for their application based on the problem's constraints.

Another model that greatly reduces computation cost while maintaining accuracy with very limited computational budget was proposed by Zhang et al[10] called ShuffleNet.This is achieved by introducing two new operations: pointwise group convolution and channel shuffle. ShuffleNet demonstrated superior performance on the ImageNet classification task, achieving lower top-1 error (absolute 7.8%) than the MobileNet [9].

3 Dataset and Setup

3.1 Original Dataset

The MNIST database [1] consists of 70,000 images of hand-written digits where the goal is to classify which digit is present in the image. Each image has been anti-aliased to produce grayscale levels, and normalized to 28x28 pixel dimensions. Since its publication, this dataset has been widely used as a testbed for different computer vision models; however, it is mostly a solved problem with modern models achieving less than 0.30% error rates. Examples that remain difficult to classify even for humans and contribute to this error rate are shown in Figure 1.

3.2 Modified Dataset and Task

The modified MNIST Dataset used for this project adds an extra layer of complexity by creating 128x128 images with three randomly transformed digits overlaid on various noisy backgrounds, where the goal is to classify the numerically largest digit on the image. A cursory investigation of the data shows that the transformations applied to the digits include resizing and rotation. The dataset is balanced across all ten classes of digits and contains a training set of 50,000 images, and a withheld test set of 10,000 images. Examples from the dataset are shown in Figure X.

3.3 Preprocessing

The grayscale values of each image were normalized by dividing by 256, so as to ensure that the input into our model are bounded between 0 and 1 and don't cause any instability during training. Furthermore, because the architectures we tested were designed for RGB images, we convert the single-channel grayscale images to three-channel images by duplicating them three times along the channel dimension. Lastly, in our data augmentation experiments, we apply various transforms which are described in Section 4.3.

4 Proposed Approach

In order to maximize our performance on the modified MNIST task, we decided to leverage modern deep learning and computer vision techniques. However, a significant challenge is the large number of dimensions along which design decisions can be made. This is further complicated by the fact that deep learning experiments are costly in terms of time and resources.

To address this, we ran controlled experiments to investigate the effect of design decisions of model architectures; model sizes; data augmentation; image resizing; and model pretraining. We maintained certain experiment-invariant hyperparameters, including:

- Stratified train/validation split of 90% to ensure representative validation set results;
- Adam optimizer to leverage adaptive learning-rates and momentum and reduce the need for optimizer hyperparameter search;
- A batch size of 256 (128 for larger models) to speed up training with available VRAM;

4.1 Model Architecture

We first ran a simple 2-layer CNN which performs well on the original MNIST dataset as a benchmark and obtained <30% validation accuracy. Therefore, we explored more advanced model architectures available through torchvision.models[1] and described in Section 2.0. These include ResNet50; MobileNet; ShuffleNet; and were chosen to explore the effect of different model architecture components such as residual connections, depth-wise separable convolutions, and point wise group convolutions. These experiments were ran with the data augmentations described in Section 4.3.

4.2 Model Size

We then explored the effect of model size by comparing results for ResNet18; ResNet50; ResNet101; and WideResNet50. While the ResNet architectures were useful in exploring the effect of model depth, the WideResNet provided some insight on the effect of model width.

4.3 Data Augmentation

One of the advantages of CNN layers in the context of computer vision is that they provide translational invariance. However, they are not invariant to other transformations such as rotation, skew, resizing, etc; which are present in the dataset. In order to potentially train a model that is more robust to these transformation, we applied data augmentation techniques using the fastai library[2]. These include jitter, perspective warp, squish, skew, tilt, totate, zoom, lighting, and contrast. Each transform is applied randomly at training, but not at validation. We evaluate the effect of data augmentation on validation performance with ResNet18.

4.4 Image Resizing

We investigate the effect of image resizing by comparing the validation performance and training time of ResNet18 after resizing from 128x128 to 64x64 and 256x256. These experiments were done with data augmentation.

4.5 Model Pretraining

We investigate the effect of model pretraining by comparing the validation performance of ResNet18 with and without pre-training on imagenet. While we were originally going to run pretraining on the MNIST dataset, we decided not to due to worries of data leakage.These experiments were done with data augmentation and without image resizing.

5 Results

5.1 Model Architecture

We find that ResNet architectures significantly outperform all other architectures. In particular, we find that WideResNet50 outperforms ResNet101 despite its reduced depth. Validation results are better than training results due to the randomized data augmentation described in Section 4.3.

We show the training loss (figure a) and validation loss (figure b) above. Per-class validation metrics for the same epoch are shown in Figure 3.

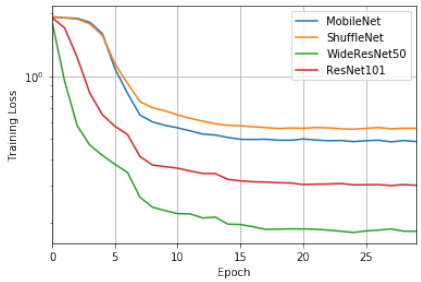
5.2 Model Size

We find that model performance increases with model size for ResNet architectures; but the returns are diminishing. We also observe that, without data augmentation, overfitting occurs after only 8 epochs and that smaller models are more susceptible to it.

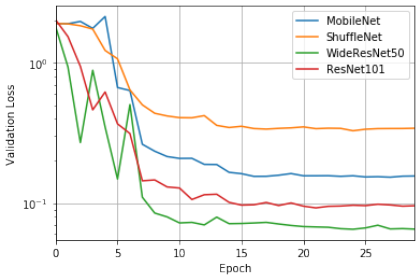
We show the training loss (figure a) and validation loss (figure b) above. Per-class validation metrics for the same epoch are shown in Figure 5.

[1] https://pytorch.org/docs/stable/torchvision/models.html

[2] https://docs.fast.ai/vision.transform.html



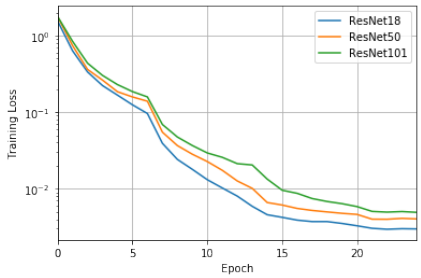
(a) Training loss for different model architectures



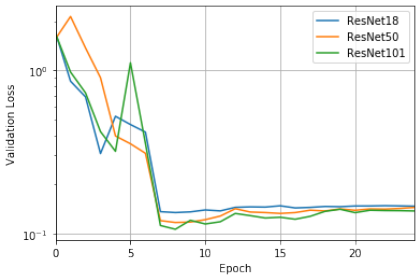
(b) valtidation loss for different model architectures

	MobileNet			ShuffleNet			WideResNet50			ResNet101		
Digit	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
0	0.000	0.000	0.000	0.000	0.000	0.000	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000
1	0.800	0.800	0.800	**1.000**	0.800	0.889	**1.000**	**1.000**	**1.000**	0.833	1.000	0.909
2	0.750	0.818	0.783	0.500	0.273	0.353	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
3	0.850	0.850	0.850	0.577	0.750	0.652	0.950	0.950	0.950	**1.000**	**1.000**	**1.000**
4	0.938	0.938	0.938	0.848	0.875	0.862	**0.939**	**0.969**	**0.954**	**0.939**	**0.969**	**0.954**
5	0.927	0.884	0.905	0.923	0.837	0.878	0.955	**0.977**	0.966	0.977	**0.977**	**0.977**
6	0.968	0.968	0.968	0.951	0.935	0.943	**1.000**	**1.000**	**1.000**	0.984	0.968	0.976
7	0.988	0.955	0.971	0.952	0.888	0.919	0.967	0.989	0.978	**0.989**	**0.989**	**0.989**
8	0.946	**0.991**	0.968	0.880	0.963	0.920	**0.981**	0.972	**0.977**	0.963	0.981	0.972
9	0.970	0.970	0.970	0.919	0.926	0.923	**1.000**	**0.978**	**0.989**	0.992	**0.978**	0.985

Figure 3: Precision, Recall, and F1 Scores for different model architectures



(a) Training loss for different model sizes



(b) Validation loss for different model sizes

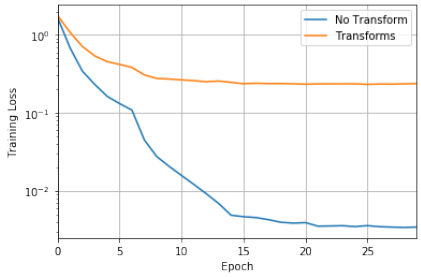
	ResNet18			ResNet50			ResNet101		
Digit	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
0	**1.000**	0.400	0.571	**1.000**	0.400	0.571	**1.000**	**0.800**	**0.889**
1	0.891	0.953	0.921	0.930	0.930	0.930	**0.956**	**1.000**	**0.977**
2	0.902	0.893	0.898	0.932	0.932	0.932	**0.925**	**0.961**	**0.943**
3	0.944	0.930	0.937	0.936	0.945	0.940	**0.974**	**0.945**	**0.959**
4	0.938	0.953	0.945	0.948	**0.965**	**0.956**	**0.950**	0.962	**0.956**
5	0.957	0.955	0.956	0.958	**0.969**	0.964	**0.967**	0.965	**0.966**
6	0.955	0.966	0.961	0.972	0.961	0.967	**0.974**	**0.977**	**0.976**
7	0.961	0.956	0.959	0.973	0.967	0.970	**0.975**	**0.974**	**0.975**
8	0.978	0.974	0.976	0.977	**0.979**	0.978	**0.984**	0.976	**0.980**
9	0.977	0.979	0.978	**0.980**	0.979	**0.980**	0.978	**0.982**	**0.980**

Figure 5: Precision, Recall, and F1 Scores for Different Model Sizes

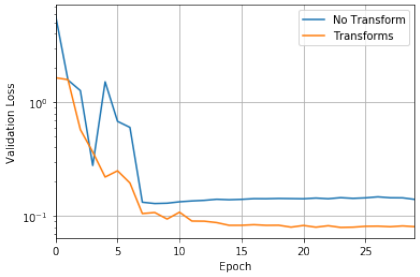
5.3 Data Augmentation

We find that model performance increases significantly with data augmentation through image transforms. We also observe that data augmentation seems to prevent overfitting.

We show the training loss (figure a) and validation loss (figure b) above. Per-class validation metrics for the same epoch are shown in Figure 7.



(a) Training loss



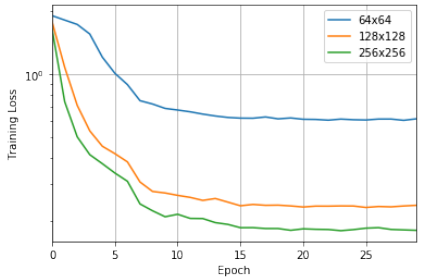
(b) Validation loss

	No Transform			Transforms		
Digit	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
0	**0.600**	**0.600**	**0.600**	0.000	0.000	0.000
1	0.894	**0.977**	**0.933**	**0.857**	**0.977**	0.913
2	0.899	0.951	0.925	**0.903**	**0.990**	**0.944**
3	0.964	0.935	0.949	**0.965**	**0.965**	**0.965**
4	**0.948**	**0.981**	**0.964**	**0.948**	0.978	0.963
5	0.957	0.939	0.947	**0.962**	**0.967**	**0.965**
6	0.960	0.977	0.969	**0.970**	**0.984**	**0.977**
7	0.962	0.968	0.965	**0.977**	**0.970**	**0.973**
8	0.986	0.971	0.978	**0.990**	**0.977**	**0.984**
9	0.978	0.973	0.975	**0.989**	**0.981**	**0.985**

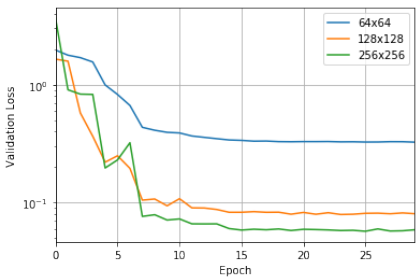
Figure 7: Precision, Recall, and F1 Scores for with/without transform

5.4 Image Resizing

We find that increasing image size significantly improves performance. However, the required computational resources also increase drastically with image size, quadrupling training time and memory consumption for a 2x increase in size.



(a) Training loss



(b) Validation loss

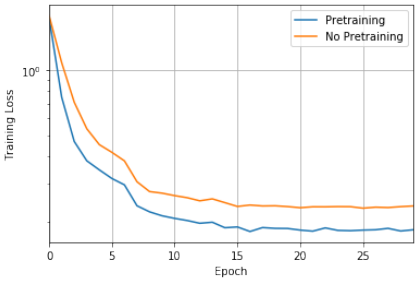
Image Size	64x64			128x128			256x256		
Digit	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
0	0.000	0.000	0.000	0.000	0.000	0.000	**0.750**	**0.600**	**0.667**
1	0.733	0.256	0.379	0.857	0.977	0.913	**0.956**	**1.000**	**0.977**
2	0.541	0.515	0.527	0.903	0.990	0.944	**0.944**	**0.990**	**0.967**
3	0.678	0.810	0.738	**0.965**	0.965	0.965	0.961	**0.980**	**0.970**
4	0.841	0.868	0.854	0.948	0.978	0.963	**0.975**	**0.984**	**0.980**
5	0.866	0.853	0.860	0.962	0.967	0.965	**0.981**	**0.974**	**0.977**
6	0.918	0.901	0.910	0.970	0.984	0.977	**0.984**	**0.985**	**0.985**
7	0.895	0.918	0.906	0.977	0.970	0.973	**0.990**	**0.988**	**0.989**
8	0.924	0.930	0.927	0.990	0.977	0.984	**0.994**	**0.987**	**0.991**
9	0.948	0.931	0.939	0.989	0.981	0.985	**0.989**	**0.988**	**0.988**

Figure 9: Precision, Recall, and F1 Scores for different image sizes

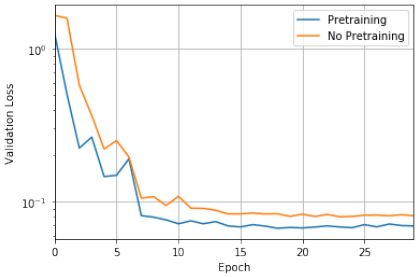
We show the training loss (figure a) and validation loss (figure b) above. Per-class validation metrics for the same epoch are shown in Figure 9.

5.5 Pre-training

We find that using model weights pretrained on ImageNet improved performance with no additional required computational resources.



(a) Training loss



(b) Validation loss

	Pretraining			No Pretraining		
Digit	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
0	**0.800**	**0.800**	**0.800**	0.000	0.000	0.000
1	**0.977**	**0.977**	**0.977**	0.857	**0.977**	0.913
2	**0.953**	**0.990**	**0.971**	0.903	**0.990**	0.944
3	**0.975**	0.960	**0.967**	0.965	**0.965**	0.965
4	**0.975**	0.975	**0.975**	0.948	**0.978**	0.963
5	**0.981**	**0.972**	**0.976**	0.962	0.967	0.965
6	**0.973**	**0.984**	**0.978**	0.970	**0.984**	0.977
7	**0.983**	**0.982**	**0.983**	0.977	0.970	0.973
8	0.989	**0.988**	**0.988**	**0.990**	0.977	0.984
9	**0.989**	**0.987**	**0.988**	**0.989**	0.981	0.985

Figure 11: Precision, Recall, and F1 Scores for different image sizes

We show the training loss (figure a) and validation loss (figure b) above. Per-class validation metrics for the same epoch are shown in Figure 11.

5.6 Final Submission

Our final submission to the Kaggle competition was the WideResNet50 with data augmentation from section 5.1, which obtained 97.7% accuracy. We were not able to add pretraining and image upscaling due to time constraints.

6 Discussion and Conclusion

Of all the architectures that we tested, ResNet architectures performed best; suggesting that residual connections can significantly increase model performance by allowing greater model depths. Furthermore, we found WideResNet50 to outperform ResNet101; this might be due to the increased layer width as claimed by original authors, however the fact that ResNet101 has 42,520,650 trainable parameters, while WideResNet50 has 66,854,730 trainable parameters might also play a role.

To better isolate the effect of model size (number of trainable parameters) on performance, we removed confounding factors of architecture design and evaluated different sizes of ResNet. We found that model performance does indeed increase with model size. Unexpectedly, we also found that larger models had larger training losses after validation convergence; suggesting they might be less prone to overfitting.

We then evaluated the effect of data augmentation through randomized image transforms. Our results show that this approach can increase the performance of smaller models like ResNet18 beyond that of larger models such as ResNet101 while also providing a strong regularizing effect that prevents

overfitting. The random nature of these transforms presumably prevents the model from simply memorizing the dataset, and allows it to learn robust representations that are more invariant to the spatial transformations applied.

We also studied the impact of image resizing and found upscaling to significantly improve performance, at the expense of a quadratic increase in required computational resources. The underlying reason for this improved performance might be related to the ability of the first convolutional layers to learn finer-grained patterns that help disambiguate otherwise difficult-to-classify examples.

Lastly, we evaluate the effect of model pretraining on ImageNet and find it to improve model performance. This was surprising since we expected the features learned for ImageNet to not be useful for the modified MNIST task, since character recognition is a much simpler task than object recognition. We expect that we could further increase performance by pretraining on the original MNIST dataset, but there is a chance of data leakage depending on which digits appear in the test set.

In the end, due to computational constraints, we couldn't run our optimal model configuration which would've combined the best features from each of our experiment: WideResNet50 with data augmentations, image upscaling to 256x256 and pretraining on ImageNet. Our final submission was WideResNet50 with data augmentation, which obtained a test set accuracy of 97.7%. For future experiments, we would like to explore pretrained model fine-tuning with gradual layer unfreezing as well as visualization of misclassified examples and model attention to identify failure mechanisms of our model.

7 Statement of Contributions

Andrei: code, experiments, figures, report: introduction/approach/dataset/results/discussion;
Ashray: overleaf editing, experiments;
Hamza: overleaf editing, report: related work;

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[2] Deng, Jia, et al. "ImageNet: A large-scale hierarchical image database."computer vision and pattern recognition(2009): 248-255.

[3] Goodfellow, Ian J., et al. "Maxout Networks."international confer-ence on machine learning(2013).

[4]K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition,"Multimed. Tools Appl., vol. 77, no. 9, pp. 10437–10453, Dec. 2015.

[5]D. H. Hubel and T. N. Wiesel, "Receptive fields, binocularinteraction, and functional architecture in the cat's visual cortex,"The Journal of Physiology,vol.160,pp.106–154,1962.

[6] K. Fukushima, "Neocognitron: a self-organizing neural net-work model for a mechanism of pattern recognition unaffected by shift in position,"Biological Cybernetics,vol.36,no.4,pp.193–202, 1980.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition,"Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2323, 1998.

[8] M.Tygert, J.Bruna, S.Chintala,Y.LeCun, S.Piantino,and A.Szlam, "A mathematical motivation for complex-valued convolutional networks,"Neural Computation,vol.28,no.5,pp.815–825, 2016

[9] . G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang,T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision appli-cations. arXiv:1704.04861, 2017.

[10]. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," inProc. IEEEInt. Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 6848–6856.