Applied Machine Learning - Mini Project 3

Nishant Mishra and Shubham Chopra

McGill University

Group 30

ABSTRACT

Aims. We analyze different Machine Learning models to process a modified version of the MNIST dataset and develop a supervised classification model that can predict the number with the largest numeric value that is present in an Image.
Methods. We experimented numerous models with different configurations for this task. The models chosen were primarily pre-trained complex neural network models, such as ResNets, VGGNets and EfficientNets. After fine-tuning the best performing models' hyper-parameters, to further boost the classification accuracy, we used various data augmentation techniques, including Affine Transformation Mappings, Scale-Space blurring, Contrast changes and Perspective transforms. By doing so, we were able to gain a higher accuracy on the test set, as compared to before data augmentation.
Results. The final fine-tuned model was able to gain an accuracy of 99% on the validation data, and an accuracy of 99.166% on the test data in the public leaderboard of the competition.

Key words. **Digital Image Processing – Computer Vision – Deep Learning – Multi-Layer Perceptrons – Convolutional Neural Networks – ResNET – VGGNET – EfficientNets – Transfer Learning**

1. Introduction

We analyze Images from a modified version of the MNIST dataset (**Yann Le Cunn**, 2001[2]), and develop a classification model to predict the number with the largest numeric value in an Image. MNIST is a dataset that contains handwritten numeric digits from 0-9 and the goal is to classify which digit is present in an image. The dataset that we work with is a modified version of the original MNIST dataset that contains 3 hand-written digits per image, overlaid on a non-white background. This problem can be formulated as a type of Image Recognition problem, which is quite well-known in the Computer Vision (CV) literature. Image Recognition is a computational approach for automated recognition of patterns and regularities in data.

For this dataset, we analyze various different models for improving the classification accuracy, including Deep Learning Models, such as versions of ResNET (**K. He et al.**, 2017[11]), VGGNET (**K. Simoyan et al.**, 2015[10]) and EfficientNet (**M.Tan and Q.V.Le**, 2019[12]). We experiment with both training the models from scratch, as well as fine-tuning pre-trained weights from the ImageNet dataset.

We also compare the accuracy of these models for different Pre-Processing Steps, including Grey-Scale to RGB conversion, re-scaling and data augmentation (set of affine transformations, blurring at different scales, Contrast adjustments and Perspective transforms).

1.1. Preliminaries

1.1.1. Neural Networks

Neural Networks are a collection of connected multi-layer perceptrons, which loosely model the neurons in a biological brain. Each perceptron, like the synapses in a biological brain, can transmit a signal to other neurons. Each perceptron that receives a signal, processes it and updates the activation signals for the neurons connected to it.

1.1.2. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of Deep Neural Networks, most commonly applied to analyzing visual imagery, in the field of Deep Learning. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting to the dataset.
The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Convolution is a Linear operation, defined between 2 signals as:-

$$f(t) * g(t) = \sum_{n=-\infty}^{\infty} f(T)g(t-T)$$

, in the discrete scenario. For a 2D Image, Convolution of the Image I(x, y) with a mask (also called a kernel) is given by:-

$$I(x,y) * g(x,y) = \sum_{n=-\infty}^{\infty} I(i,j)g(x-i,y-j).$$

In general, a CNN has the following layers:-

1.1.2.1 *Convolutional Layer*
The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, and extend to the

full depth of the input. During the forward pass, each filter is convolved across the width and height of the input volume, peforming a convolution between the image and the kernel, thus producing a 2-dimensional activation map of that filter. In this process, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

Three hyper-parameters control the size of the output volume of the convolutional layer: the depth, stride and zero-padding. The spatial size of the output volume can be computed as a function of the input volume size **W**, the kernel field size of the convolutional layer neurons **K**, the stride with which they are applied **S**, and the amount of zero padding **P** used on the border. The formula for calculating how many neurons "fit" in a given volume is given by:-

$$\frac{W - K + 2P}{S} + 1$$

1.1.2.2 *Pooling Layer*
Pooling is an important concept, and is a type of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling being the most widely used. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. Pooling relates to feature extraction at different Scales of the Image, and has roots in Scale-Space Theory.

1.1.2.3 *Non-Linear Activation Layer*
Applying a Activation function like the hyperbolic tangent, sigmoid or ReLU is an important step in the process to generate non-linearlity in the network. It increases the non-linear behaviour of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

1.1.2.4 *Fully Connected Layer*
After several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all neurons in the previous layer, as the concept in regular artificial neural networks (ANNs).

2. Related Work

LeCun, Y., et al.[15], first describe a Neural Network to classify Handwritten digits on a dataset that was the precedent to the MNIST. Image Classification has come a long way since then. **Simonyan et al.**[10] describe VGGNet, a block based CNN architecture that achieved then state-of-the-art accuracies on the ImageNet (**O. Russakovsky et al.**)[13] dataset. Further improving on this, (**K. He et al.**)[11] introduced ResNet, a skip-layer based CNN architecture that used skip connections to jump over layers, allowing for the usage of much deeper neural networks and solving the problem of vanishing and exploding gradients. ResNet was considered to achieve better performance accuracies on ImageNet, as compared to even humans, i.e., super-human performance.

More recently, the community has been interested in more efficient Deep Learning architecures for Image Classification. (**M. Tan and Q.Le**, 2019[12]) describe a novel scaling method for CNNs, that allows for much more efficient CNNs, while achieving even better performance on the ImageNet dataset than ResNet.

3. Dataset and Preparation

The given dataset contains 50,000 modified MNIST images.The images are grayscale images of size 128*128. Each image contains three MNIST style randomly sampled numbers on custom grayscale backgrounds each at various positions and orientations in the image. The task was to train a model in order to identify the number of highest value in the image.

3.1. Data Preparation

In order to make the data compatible for training with different models using different training strategies, it needs to be preprocessed. In our case, we tried training models from scratch on the grayscale images as well as fine tuning models pretrained on larger RGB dataset. In the former case the preprocessing only involved mean normalizing the image intensities before passing them for training.But, since most of the pretrained models available for finetuning have been trained for RGB images in the Imagenet dataset. Therefore in order to use our data with the transfer learning approach,we had to resize the images in our data from 128*128 to 224*224. Also since images in the dataset were 1 dimensional grayscale images, we needed to convert them into three dimensional images for them to be meaningful data for the pretrained models. For this we just repeated the images in all three dimensions to give an impression of RGB data. Ofcourse the normalization step was still applicable. We split out dataset into a 80-20 training and development data split for training.

3.2. Data Augmentation

Data Augmentation is a strategy to increase the size and diversity of our data to enable the models to learn better representations and generalize better. Its mostly an artificial method where spatial and transformations are applied to the image to create slightly transformed alternate versions of the Image. There are a number of transformations that can be applied such as Affine Transformations(scaling, rotation,translation,shear etc) , Introducing noise, contrast changes, blurring, perspective transforms etc. This strategy has been proved to be really effective as mentioned in (**Luis Perez, Jason Wang, 2017**)[17]. For certain trials of our models we also augmented our data to improve the performance and it had a huge impact on our model's performance as mentioned in the results section. Some examples of Data augmentation we applied are shown in ***Figure 1.***

4. Proposed Approach

We trained and evaluated several models, which are quite well-known in the MachineLearning and Deep Learning Literature including VGGNet (**K. Simoyan et al., 2015**)[10], ResNet (**K. He et al., 2017**)[11] and EfficientNet (**M. Tan and Q.Le, 2019**)[12]. We used the Transfer Learning approach to fine tune models pretrained on ImageNet to our dataset.

4.1. Residual Neural Networks (ResNet)

ResNet is an Artificial Neural Network (ANN) that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks utilize skip connections, or short-cuts to jump over some layers. This allows for much deeper-layered networks. One motivation for skipping over layers is to avoid the problem of vanishing gradients, by reusing activations from

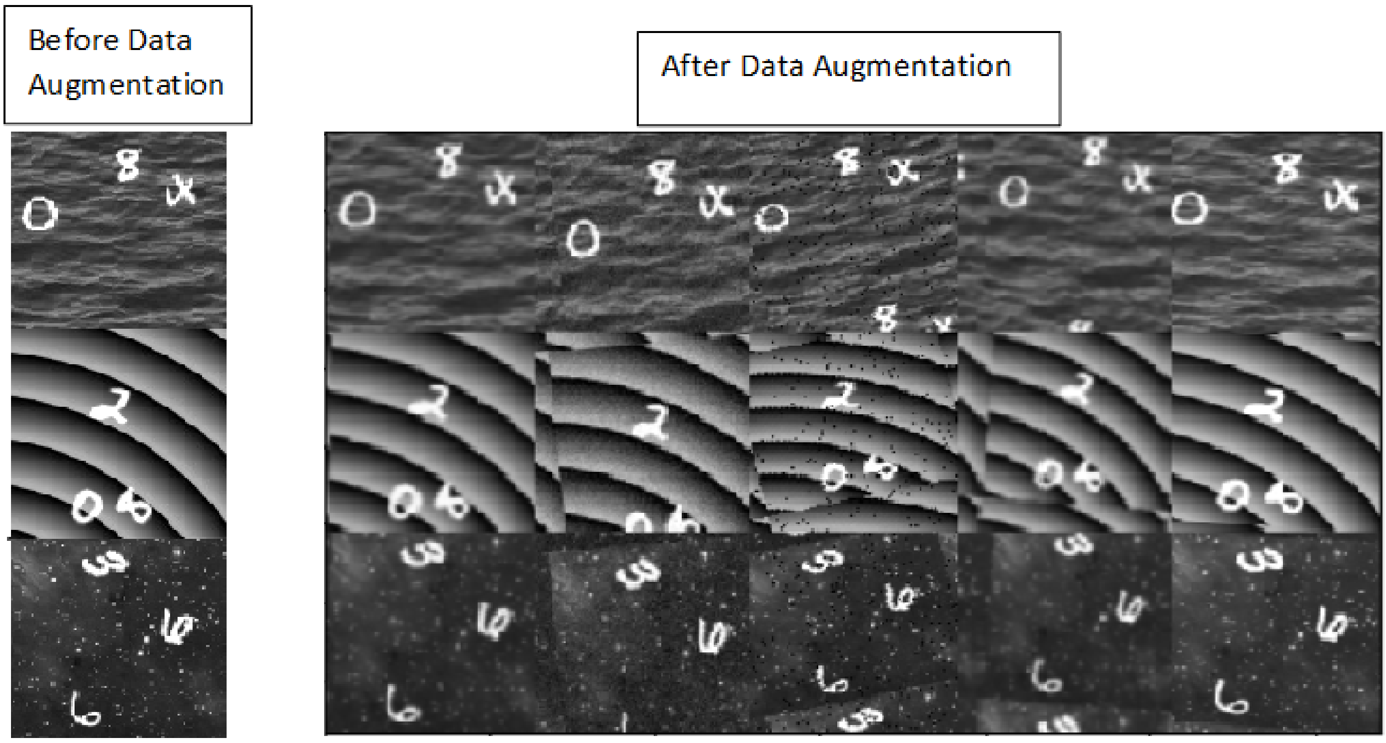

Fig. 1. Example of Data Augmentation applied to our data. Includes Blurring,translation,shear,adding Noise etc

a previous layer until the adjacent layer learns its weights. During training, the weights adapt to mute the upstream layer, and amplify the previously-skipped layer. In the simplest case, only the weights for the adjacent layer's connection are adapted, with no explicit weights for the upstream layer. This works best when a single non-linear layer is stepped over, or when the intermediate layers are all linear. Skipping effectively simplifies the network, using fewer layers in the initial training stages. This speeds learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through. The network then gradually restores the skipped layers as it learns the feature space. Typical ResNet models are implemented with double- or triple-layer skips that contain nonlinearities (ReLU) and batch normalization in between.

4.1.1. Forward Propagation

For single skips, the layers may be indexed either as l-2 to l or l to l+2. Given a weight matrix $W^{l-1,l}$ for connection weights from layer l-2 to l, and a weight matrix $W^{l-2,l}$ for connection weights from layer l-2 to l, then the forward propagation through the activation function would be:-

$$a^l = g(W^{l-1,l}.a^{l-1} + b^l + W^{l-2,l}.a^{l-2})$$
$$= g(Z^l + W^{l-2,l}.a^{l-2})$$

where,
a^l = activations (outputs) of neurons in layer l,
g = the activation function for layer l,
$W^{l-1,l}$ = the weight matrix for neurons between layer l - 1 and l,
$Z^l = W^{l-1,l}.a^{l-1} + b^l$

In the absence of an explicit matrix $W^{l-2,l}$, forward propagation through the activation function simplifies to:-

$$a^l := g(Z^l + a^{l-2})$$

4.1.2. Backward Propagation

During back-propagation in ResNets, learning for the normal path is given as :-

$$\delta w^{l-1,l} := -\alpha \frac{\partial E^l}{\partial w^{l-1,l}} = -\alpha a^{l-1}.\delta^l$$

, and for Skip paths:-

$$\delta w^{l-2,l} := -\alpha \frac{\partial E^l}{\partial w^{l-2,l}} = -\alpha a^{l-2}.\delta^l$$

where, α = learning rate of neurons at layer l, δ^l = the error signal of neurons at layer l, a^l = the activation of neurons at layer l.

In general, for "K" skip path weight matrices-

$$\delta w^{l-k,l} := -\alpha \frac{\partial E^l}{\partial w^{l-k,l}} = -\alpha a^{l-k}.\delta^l$$

In ResNet, as the learning rules are similar, the weight matrices can be merged and learned in the same step.

4.2. VGGNet

VGGNet is a CNN architecture widely used for Image Classification, first described by the Visual Geometry Group (VGG) at Oxford University. VGGNet divides the CNN into basic VGG blocks, where each VGG block consists of a sequence of convolutional layers, followed by a max pooling layer for spatial downsampling. This network is characterized by its simplicity, using only 33 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier. There are also different variants of VGGNet, including VGG16 and VGG19, where 16 and 19 describe the number of weight layers in the network.

Table 1. Accuracy of Models under different scenarios

Model	**Data Augmentation**	**Pretrained**	**Validation Accuracy**	**Kaggle Accuracy**
VGGNet 19	✗	✗	95.16%	94.9%
	✓	✓	97.8%	97.36%
ResNet18	✗	✗	95.6%	95.33%
	✓	✓	98.3%	97.9%
ResNet50	✗	✗	97.5%	96.9%
	✗	✓	98.1%	98.56%
	✓	✓	99%	99.166%
EfficientNet_b5	✓	✓	99.08%	98.73%

4.3. EfficientNet

EfficientNet proposes a novel model scaling method (referred to as Compound Scaling in the original paper) that uses a simple yet highly effective compound coefficient to scale up CNNs in a more structured manner. Unlike conventional approaches that arbitrarily scale network dimensions, such as width, depth and resolution, this method uniformly scales each dimension with a fixed set of scaling coefficients. This results in upto 10x better efficiency in models, and superpasses the state-of-the-art accuracy in ImageNet.

4.4. Transfer Learning

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned. Thus, it allows for weights that have been learned in one setting, to be exploited to improve generalization in another setting. In transfer learning, we first train a base network on a base dataset and task, and then we fine-tune the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.

5. Results and Discussions

5.1. Analysis of different Classifiers

As we can see in *Table 1*,we tried out several different CNN architectures under a number of settings. Resnet50 performed the best on kaggle public leaderboard with 99.16% while EfficientNet,although having a superior validation accuracy on our development set only reached 99% accuracy on the kaggle test data. Overall ResNet 18 is the most efficient of all the models in terms of time of convergence to accuracy tradeoff, While VGG19 was the least efficient among them having thrice as many parameters as ResNet50 and taking much longer to converge.

5.2. Hyperparameters

We tested different hyperparameter settings while training the models and tried to come up with a consistent setting for the different parameters. Train-Test data split ratio, number of epochs, learning rate, choice of optimization algorithm, learning rate scheduler, etc were all experimented with. We got the best performance using a SGD optimizer with an initial learning rate of 0.01 and a learning rate scheduler that scaled the learning rate on a plateau. This particular hyperparameter setting when used with a ResNet50 on a 80:20 data split and trained for 40 epochs yielded our best performance of 99.166% on the kaggle public leaderboard.

5.2.1. Transfer Learning and Data Augmentation

Initially we trained the ResNet18, ResNet50 and VGG19 models from scratch on the given grayscale data. The accuracy for them got maxed out at near 96%. So, instead of just training the models from scratch,we used the transfer learning based approach of finetuning versions of these models pretrained on Imagenet dataset. This approach resulted in a large impact on the performance of the order of as high as 2% improvement in accuracy. In both the transfer learning approach as well as from-scratch training, we observed that the validation loss plateaued even though the training loss kept decreasing. It was clear the model was overfitting. Therefore we augmented the training data in order to make the models more generic. We used the ***imgaug*** library to introduce a number of image transformations to generate several times more data. We performed scaling, rotation, blurring, edge detection and overlay to the images to produce new images. The strategy to implement data augmentation resulted in significant gains in both fine-tuned models and ones trained from scratch as can be seen from the Table 1.

5.3. Models used for Kaggle Submission

We first used the models trained on the data from scratch, which reached an accuracy of 95.56% on kaggle. Then we used pre-trained models for increasingly advanced architectures, of which ResNet50 performed the best when trained on augmented training data reaching a development set accuracy of 99% and a score of 99.166% on the public leaderboard of Kaggle.

6. Conclusions

1. Transfer Learning approaches for Fine-tuning the models pre-trained on ImageNet provided a huge boost in accuracy. It allowed for weights used in ImageNet, to be exploited to improve generalization in our dataset.
2. Data Augmentation techniques provided a huge boost in the Kaggle Test accuracy. In the absence of data augmentation methods, validation loss plateaued, while training loss continued decreasing.
3. ResNet50 performed the best among all other models. ResNet50 with data augmentation and fine-tuning pre-trained weights provided the best accuracy on the Kaggle dataset of 99.166%. The performance decreased to 98.56% without data augmentation.
4. EfficientNets failed to out-perform ResNet50 on the Kaggle Test set, despite being the State-of-the-art on ImageNet.
5. Handwritten Digit Recognition (in datasets like MNIST) can be considered as a solved problem, since we were able to achieve near perfect Test accuracy in the dataset. Similar

techniques can be transferred to solve higher order problems like Scene Text Detection, Optical Character Recognition (OCR) and Handwriting Analysis.

7. Future Work

More data points should be collected. Since, most deep learning algorithms are very data intensive, it can provide a major boost to performance by collecting more data-points to augment training. ResNet-18, ResNet-34 and ResNet-50 were used in our tests. ResNet-100 can also be trained and hyper-parameter tuning experiments can be performed to further increase validation accuracy. EfficientNets did not out-perform ResNet-50 on the Kaggle Test dataset, despite performing better on the Validation set and being the State-of-the-art in ImageNet performance. We used the B5 version of EfficientNets in our tests, however, more hyper-parameter tuning and testing with different versions of EfficientNets (B3 and B7) can help increase performance. Since, data augmentation methods provided a huge boost to performance, more research can be carried out in selection of better data augmentation methods. For this, ideas and inspiration can be borrowed from the fields of Optical Character Recognition (OCR) and Handwriting Analysis.

8. Statement of Contributions

This project is a collaborative effort between two team members. The tasks were evenly distributed and every member contributed equally.

References

[1] Basheer, I.A., and Hajmeer, M., 2000. Artificial neural networks: fundamentals, computing, design, and application. Journal of microbiological methods, 43(1), pp.3-31.

[2] Yann.lecun.com., n.d. MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. [online] Available at: http://yann.lecun.com/exdb/mnist/ [Accessed 30 May 2018].

[3] Ciresan, D.C., Meier, U., Gambardella, L.M. and Schmidhuber, J., 2011, September. Convolutional neural network committees for handwritten character classification. In Document Analysis and Recognition (ICDAR), 2011 International Conference on (pp. 1135-1139). IEEE.

[4] Deng, L., 2012. The MNIST database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine, 29(6), pp.141-142.

[5] Gedeon, T.D. and Harris, D., 1992, June. Progressive image compression. In Neural Networks, 1992. IJCNN., International Joint Conference on (Vol. 4, pp. 403-407). IEEE.

[6] Gedeon, T.D., Catalan, J.A. and Jin, J., Image Compression using Shared Weights and Bidirectional Networks. In Proceedings 2nd International ICSC Symposium on Soft Computing (SOCO'97) (pp. 374-381).

[7] Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. Cats and dogs. CVPR, pp. 3498–3505, 2012.

[8] Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. arXiv preprint arXiv:1710.05941, 2018.

[9] Alexandra Chronopoulou, Christos Baziotis, Alexandros Potamianos. (NAACL-HLT, 2019). An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models.

[10] K. Simonyan, A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations, 2015

[11] Kaiming He , Xiangyu Zhang and Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, CoRR, http://arxiv.org/abs/1512.03385.

[12] Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. To appear, ICML 2019.

[13] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. CoRR, abs/1409.0575, 2014.

[14] Lin, H. and Jegelka, S. Resnet with one-neuron hidden layers is a universal approximator. NeurIPS, pp. 6172– 6181, 2018.

[15] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[16] Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML, pp. 448–456, 2015.

[17] Luis Perez, Jason Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. CVPR 2017.