ACSE-4: Project 3 - Neural network for Kuzushiji-MNIST

Team Name: Entropy
Keer Mei, Yujie Zhou, Adanna Akwataghibe, Tayfun Karaderi

May 24, 2019

Abstract

The Kuzushiji-MNIST (KMNIST) data set is a set of images containing various iterations of 10 Japanese characters. We trained LeNet and AlexNet neural networks to predict the KMNIST test set, as well as attempted at classifying the test set using PCA with a KNN algorithm. To enhance the accuracy of our models, we attempted at incorporating combinations of dropout, data augmentation, retraining, and transfer learning into our models. A model averaging approach was used to enhance the final accuracy of our best performing model.

1 Introduction

The KMNIST is similar to the MNIST dataset with 28x28 grayscale, 70,000 images, labelled into 10 classes. This data set consists of the Kuzushiji cursive Japanese characters. These characters are not used in writing or taught in schools since the modernisation of the Japanese language. However, there are classical literature written before this modernisation that cannot be read by the average Japanese. Therefore, we aim to use model approaches like LeNet, AlexNet and model averaging to classify these characters with a high enough accuracy, so that these classical books can be read.

2 Training Approach

Initially, our team decided to diversify into three separate approaches to find the optimal classifier for the KMNIST data set. We looked at supervised learning on two different neural network architectures: LeNet and AlexNet, and unsupervised learning by PCA in combination with a KNN algorithm.

The next step was focused on searching for the optimal hyper-parameters on each of the two neural networks. We investigated how weight decay, momentum, and the number of epochs would affect the final validation accuracy. Beyond the hyper-parameters search, we also introduced various combinations of dropout, transfer learning, re-training as well as data augmentation into our models to understand their impact on the prediction accuracy.

Our final investigation focused on the effects of training on a complete data set versus a training-validation split. Throughout these endeavors, we used cross-validation techniques to ensure that our models were neither over-fitting or under-fitting. On the LeNet architecture, we used a shuffle split for cross-validation while on the AlexNet architecture, we used a k-fold cross-validation.

3 Data pre-processing

Before we trained our neural networks, we pre-processed the KMNIST images by normalization. During training, the data set is enlarged using image transforms such as rotations, flips, shears, and translations. Both of these techniques were implemented using a `CustomTensorDataSet` object. We ignored transforms that included large rotations (i.e. 90 to 180 degrees), because they can alter the nature of the Japanese characters significantly and increase the likelihood of mis-classification. We assumed that the data given was of high quality, had no outliers and all the labels matched correctly with the images.

3.1 Normalization

Our approach to pre-processing of the KMNIST data is image normalization which reduces the image pixel values to a 0-mean and 1-standard deviation distribution. This is achieved by subtracting each individual pixel value by the mean of the entire data set, and dividing by the standard deviation of the entire data set.

Data normalization has the benefit of scaling the data values to a common range, which improves convergence speed. Although all of the images in the KMNIST data set are within the same pixel values between 0 to 255 and normalization may seem redundant, normalization is a best practice that has been incorporated into our approach.

3.2 Data Augmentation

transforms	features
Random Rotation	• rotation up to 10 degrees
Random Affine	• rotation up to 10 degrees • translation • shear
Random Choice	• rotation up to 10 degrees • horizontal flip • translation • shear

Table 1: Table of transforms used.

Data augmentation is a way to extend the KMNIST data set and provide neural networks with more training images in previously unseen formats. We tried a variety of approaches to data augmentation. First, we compared the effects of a once-per-epoch `RandomRotation` transform against a once-per-epoch `RandomAffine` transform on a basic LeNet5 architecture. The results of this experiment showed that the validation accuracy with the `RandomAffine`transform (98.5 % see Table 3) performed better than the `RandomRotation` transform (98.1 %). This can be explained by the `RandomAffine` transform contributing more dynamic images towards the training model as it includes three different transforms.

One key takeaway of data augmentation is that it reduces the network's tendency to over-fit on the training data. In Figure 1, the validation accuracy at every epoch is above the training accuracy which supports the conclusion that the model is under-fitting the training data. In this regard, data augmentation has been noticed to perform detrimentally in combination with dropout and weight decay, as all of these techniques combined produces an noticeably under-fitted network.

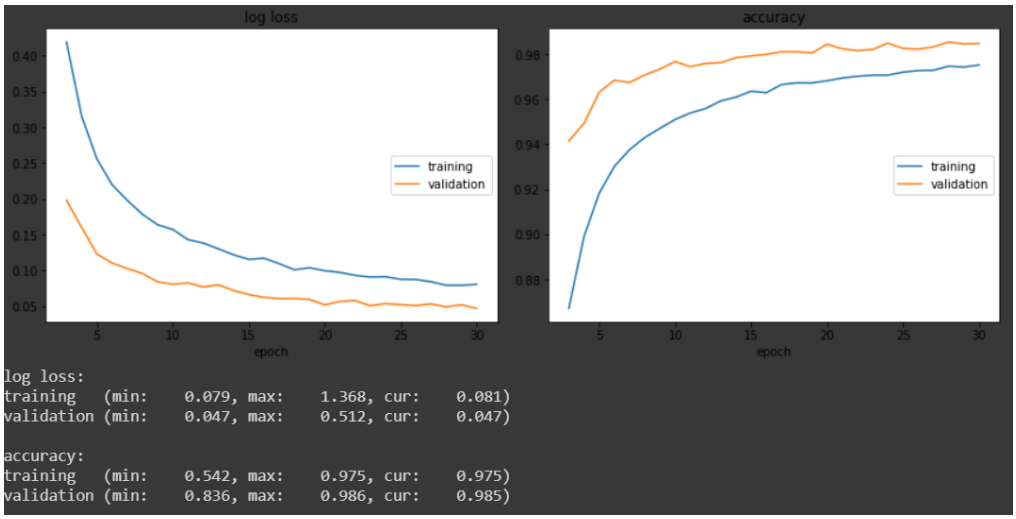

Figure 1: Training and Validation accuracy of a LeNet5 model trained with random affine transforms.

In the later models that we trained, we altered our training function to include multiple-per-epoch data augmentations. This approach is equivalent to enlarging the data set by training the network numerously through a variety of different augmented images at every epoch. To do this, we switched to a `RandomChoice` augmentation that chooses from a list of pre-selected augmentations including: rotation, translation, shear, and horizontal flip. The results of doing this multiple-per-epoch augmentation is improved validation accuracy of up to 100 % in as early as the 10^{th} epoch. The cost to adding multiple-per-epoch augmentations however, is a significantly increased computation time.

4 Neural Network Architecture

4.1 LeNet

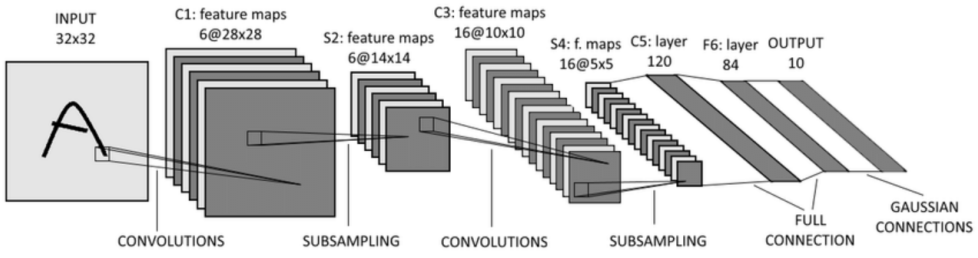

Figure 2: Basic LeNet5 architecture[1].

The basic LeNet5 neural network (Figure 2) is one of the first architectures that we attempted to train on the KMNIST data set. The simplicity of this network allows it to be easily adapted to 28x28 grayscale images. Later on, we attempted to use variations of the LeNet model including:

1. changing LeNet5 by using kernel sizes of 3 and 7 in the convolution layers C1 and C3. The Basic LeNet5 uses a kernel size of 5. This also changes the sizes of the S2 and S4 layers.
2. adding Dropout to the fully connected layers C5 and F6. This results in significant under-fitting.
3. multiplying the output of the convolution layers C1 and C3 by a factor of 3 and adding an additional fully connected layer C6 after C5.

4.1.1 LeNet with Dropout

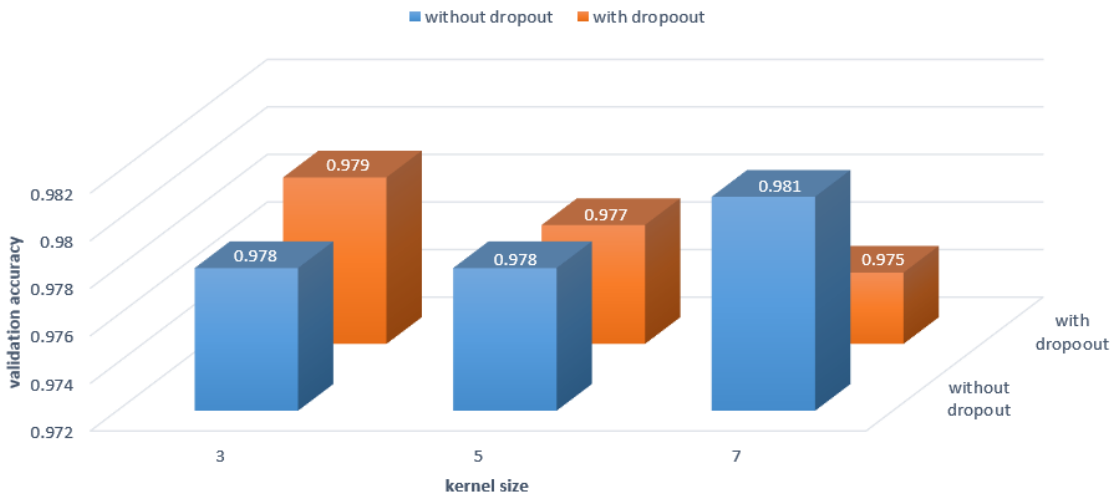

Figure 3: Validation accuracy of LeNet with different kernel sizes and dropout

Experimentation with kernel sizes and dropout (Figure 3) in combination with different weight-decays (Figure 12, Figure 13) showed that the validation accuracy can vary sporadically. It is difficult to determine exactly what is an ideal combination of kernel size, dropout, and weight-decays.

Originally, dropout was seen as a technique to reduce over-fitting. However, we noticed that dropout caused our network to under-fit the training data significantly. The implications of including drop out meant that when used in conjunction with data augmentation, our models performed worse than when they were trained without dropout.

4.1.2 Transfer Learning and Retraining

One of the first experiments we conducted was transfer learning using a model that had previously been trained on the MNIST data set. This had unfavorable results as the training accuracy did not noticeably improve while the validation accuracy became unpredictable. By their very nature, the MNIST and KMNIST data sets are very different images; the MNIST are images of numbers while the KMNIST are images of Japanese characters. It was concluded from this experiment that transfer learning from models trained on other data sets would not be a prudent approach.

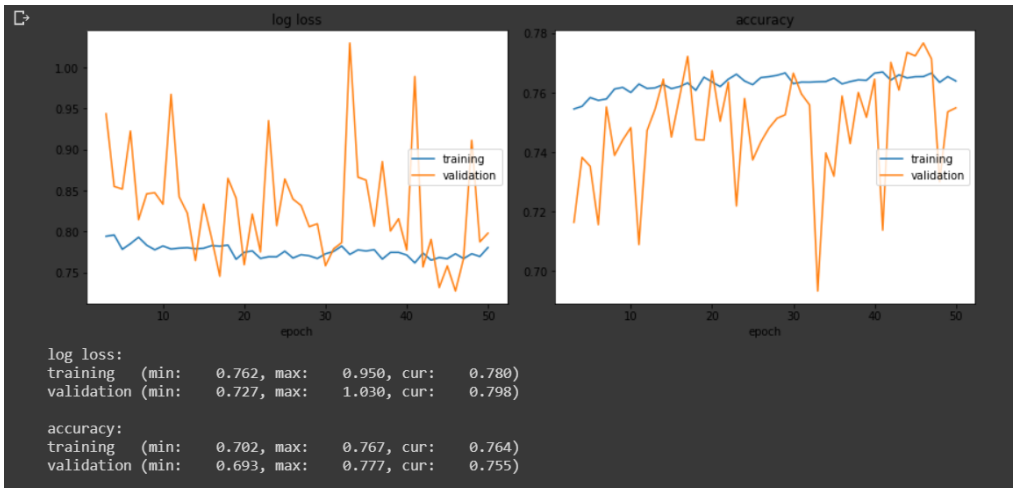

Figure 4: Transfer learning on a MNIST trained LeNet model

In contrast to transfer learning, we discovered that retraining is a more favorable approach with significant benefits to the model validation accuracy. Retraining entails training a model more than once, first on augmented data and then once on the original KMNIST data set. What this accomplishes is to allow the model to be trained on a larger sized data set. This approach forms the basis of our final competition model submissions, which were trained on a multiple-per-epoch augmentation in addition to the retraining on the original KMNIST data set.

4.1.3 LeNet with Extended Layers

In our final submission to the Kaggle competition, our team retrained a LeNet5 model using an additional fully connected layer called C6. We also increased the output of the convolution layers C1 and C3 by a factor of 3. This new architecture, in combination with data augmentation and retraining achieved our best validation accuracy of 100 % .

4.2 AlexNet

AlexNet is a CNN with 5 convolutional layers 3 max pooling and 3 fully connected layers as displayed in Figure 5. The original Alex-Net was designed for 227 by 227 images with 3 channels. The architecture was adapted to work with single channel 28x28 images. The resized AlexNet layers are shown in Table 2.

Layer	f	p	s	number of filters
conv1	3	0	1	8
MaxPool	2	0	2	8
conv2	5	2	1	16
MaxPool	3	0	2	16
conv3	3	1	1	32
conv4	2	1	1	16
conv5	2	1	1	32
MaxPool	3	0	2	32

Table 2: Adapted AlexNet Parameters.

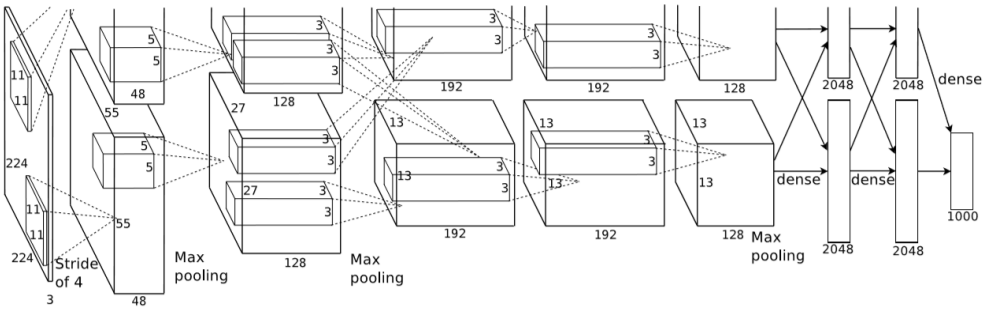

Figure 5: Basic AlexNet architecture[2].

Later on, techniques of data augmentation and hyperparameter tuning were used to reduce over-fitting and avoid under-fitting. In section 6, we discuss how AlexNet was trained in more detail.

5 Unsupervised Approach

5.1 Principal Component Analysis (PCA)

PCA is a procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. This method was used to reduce the dimension of the KMNIST images from 784 to 80 as we found this produced the best accuracy, whilst maintaining a reduced computational cost. This reduced data set was then passed to the KNN classifier.

5.2 K-Nearest Neighbours (KNN) Algorithm

KNN is a supervised learning algorithm for classification where the labeled data is used to draw decision boundaries. In the classification setting, the unseen data is classified based on its K-nearest neighbours by forming a majority vote. The L2 norm was used as our distance metric as this gave good validation accuracy scores.

Overall, a validation set accuracy of 97.7 % was obtained using this method. Alternative classification algorithms such as the Random-Forest and Naive-Bayes algorithms were also tested. However, despite hyper-parameter tuning, they resulted in much lower validation set accuracies of 92.7 % and 76.2 % respectively. All of these scores were lower than LeNet and AlexNet results.

6 Validation Approach and Hyperparameter Selection

6.1 Shuffled split

Shuffle split is a technique we used to divide the full KMNIST data set into two separate pools. We used a 90:10 split, which allocates 90 % of the full KMNIST data set into a training set and the 10 % remaining to a validation set.

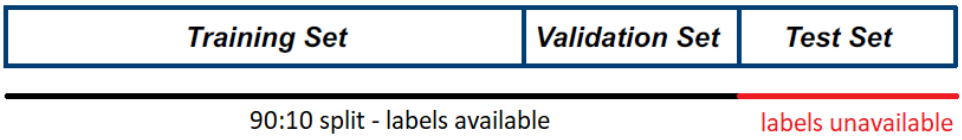

Figure 6: Shuffle split data.

While we train the networks on the training set, the validation set becomes "unseen data" that we can use to evaluate our model at each epoch. This provides a measure to gauge whether our model is improving, over-fitting, or under-fitting to the training data. Logically, the higher the validation accuracy that we obtain from our network, the more confidence we have in its performance on the actual test data set.

6.2 K-fold Cross validation

K-fold Cross validation was used find the optimal hyperparameters in AlexNet. The full training set was split into k parts of training and validation using a `k_split` function. Then the cross validation would be performed using `k_fold_optimisation`.

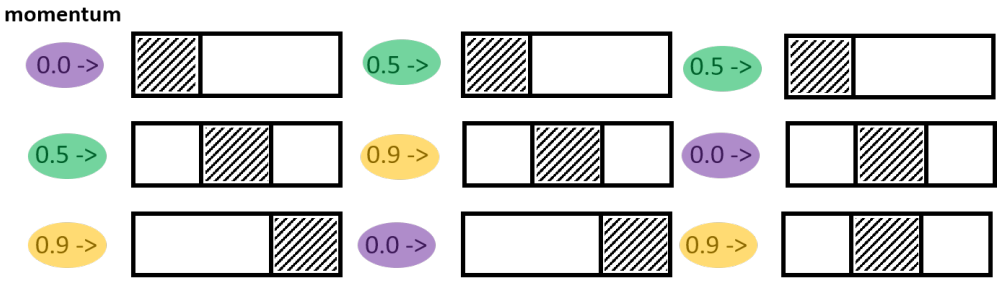

Figure 7: Example of 3-fold cross validation with momentum

For example, if we want to optimise the momentum parameter, we list the momentums to test e.g. [0.0, 0.5, 0.9], and split the training set into 3-folds of training and validation. Figure 7 above shows the model being trained repeatedly in each separate fold. For each value of momentum, an average accuracy is calculated from the scores of each fold. The momentum with the best validation accuracy is then chosen as the best parameter value.

Using the k-fold cross validation is computationally expensive, and we would resort to manually tinkering with the hyper parameter. Our results are shown in Figure 8.

6.3 AlexNet hyper-parameters search

Momentum	Weight decay	Optimiser	Data Augmentation	Drop out probability	Num. epochs	Learning rate	Max. Validation Accuracy
0.5	0	SGD	Normalise	0.5	40	1.00E-02	0.985
0.9	0	SGD	Normalise	0.5	30	1.00E-02	0.985
0.5	0	SGD	Random	0.5	50	1.00E-02	0.98
0.5	0	SGD	Random	0.1	50	1.00E-02	0.98
0.5	0	SGD	Normalise	0.1	50	1.00E-02	0.985
0.5	0	SGD	Normalise	0	50	1.00E-02	0.987
0.5	0	SGD	Normalise	0.1	50	1.00E-02	0.987
0.92	0	SGD	Normalise	0.1	50	1.00E-02	0.986
0.92	1.00E-03	SGD	Normalise	0.1	50	1.00E-02	0.988
0.95	1.00E-03	SGD	Normalise	0.1	50	1.00E-02	0.989
0.95	5.00E-03	SGD	Normalise	0.1	50	1.00E-02	0.978
0	0.00E+00	Adam	Normalise	0.5	50	1.00E-03	0.985
0	0.00E+00	Adam	Normalise	0.2	50	1.00E-03	0.988
0	0.00E+00	Adam	Random	0	50	1.00E-03	0.989
0	1.00E-03	Adam	Random	0	60	1.00E-03	0.989

Figure 8: AlexNet models features with accuracy

Initially, we used a SGD optimizer with default drop out probability of 0.5. In this case, changing the momentum did not change the validation accuracy from 98.5%. When `RandomAffine` data augmentation was added with SGD, the lowest validation accuracy was obtained at 98.0%. This signified severe under-fitting caused by adding more randomly generated images. We then removed the `RandomAffine` data augmentation and reduced the dropout probability, to get scores between 98.6% and 98.7%. We also noticed that with a low dropout rate, and momentum of 0.92, our data was over-fitting - as shown by Figure 9. Therefore, we increased the weight decay (L2 regularisation) to mitigate this, and we obtained the best AlexNet validation accuracy 98.9%.

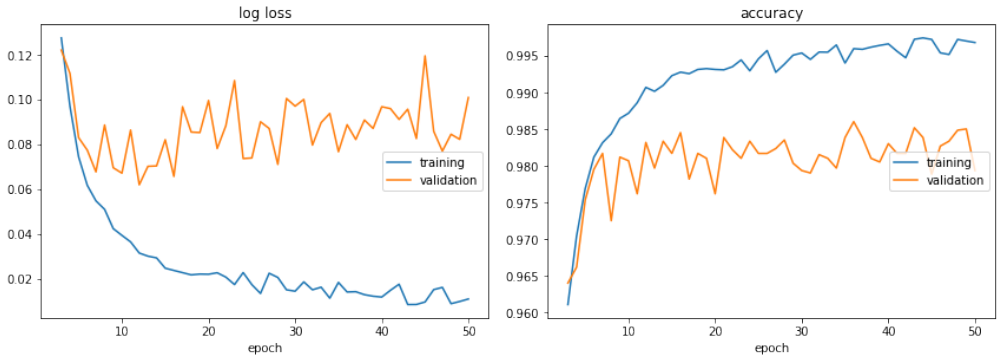

Figure 9: SGD model overfitting

We also tested using another optimizer algorithm called Adam - which combines the advantages of two SGD extensions RMSProp and AdaGrad. The best accuracy obtained (98.9%) was using Adam with `RandomAffine` data augmentation and without dropout. The optimal graph is also shown in Figure 10.

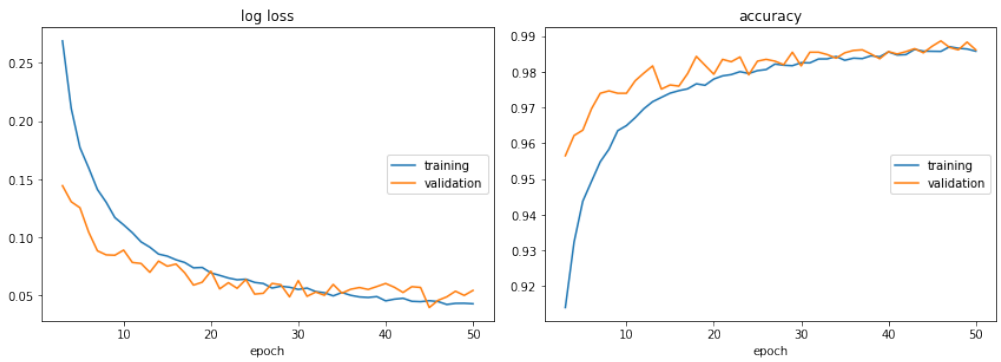

Figure 10: Adam optimizer AlexNet Parameters

The performance of AlexNet and LeNet were similar, however LeNet was able to achieve up to 100% validation accuracy, so this was included as one of our choices for the Kaggle submission. The optimal AlexNet model was used in the Model averaging (discussed below)

6.4 LeNet hyper-parameters search

The main hyper-parameter that we focused on was the weight decay value used in the SGD optimizer for training the LeNet network. Weight decay values of 0.0 , 1e-3, 1e-4, 1e-5 were investigated in combination with varying kernel sizes and dropout to check their corresponding validation accuracy.

It should be noted that the comparisons below (Figure 11, Figure 12, Figure 13) below were produced without any data augmentation, retraining or extended architecture.

Figure 11 shows that the a weight decay of 1e-3 is the best for the LeNet model with kernel size 5, irrespective of dropout.

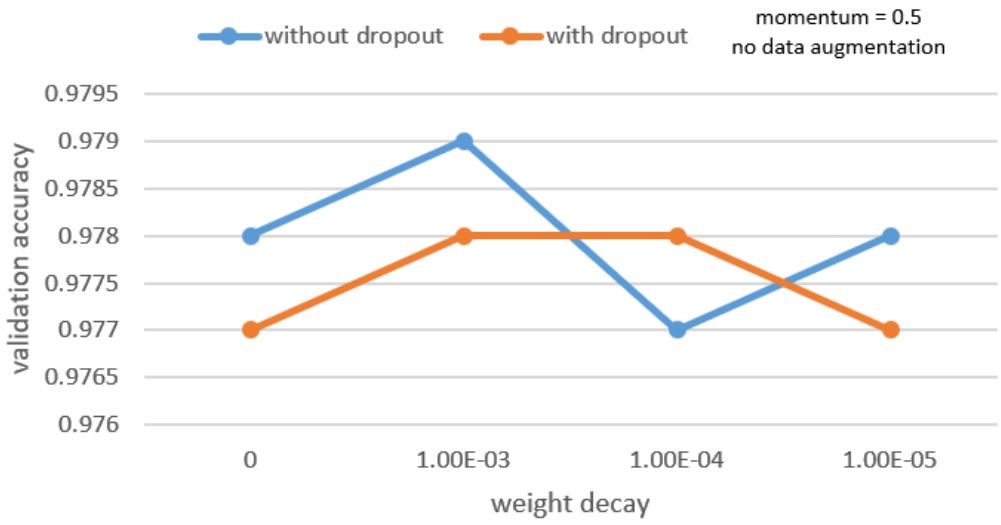

Figure 11: Comparison of LeNet5 validation accuracy.

- Without dropout
  The immediate conclusion from Figure 12 is that a kernel size of 7 obtains the highest validation accuracy,

irrespective of weight decay. However, this conclusion becomes irrelevant when dropout is included.

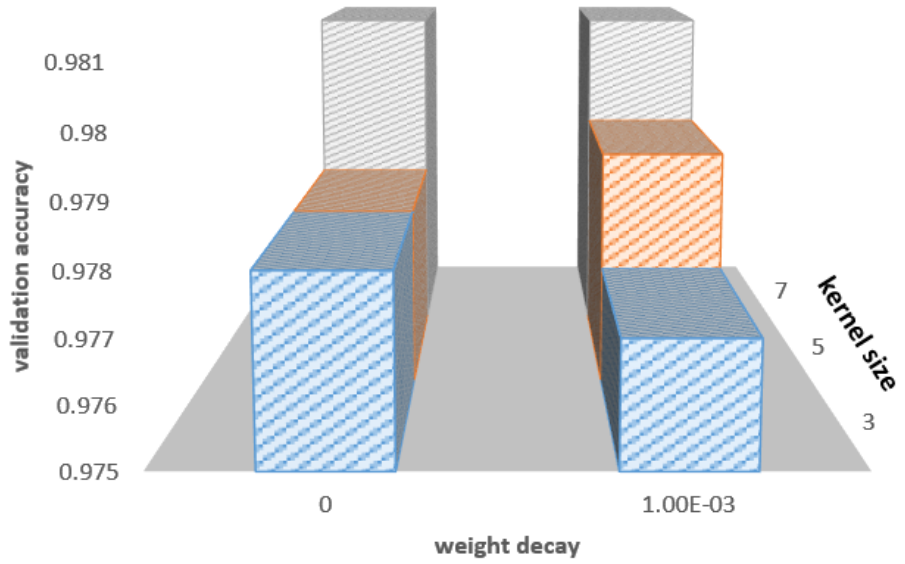

Figure 12: Comparison of validation accuracy without dropout.

- With dropout
  When dropout is included, a LeNet network with a kernel size of 3 obtains the highest validation accuracy. In fact, the highest validation accuracy of 0.982 or 98.2 % uses a weight decay value of 1e-3.

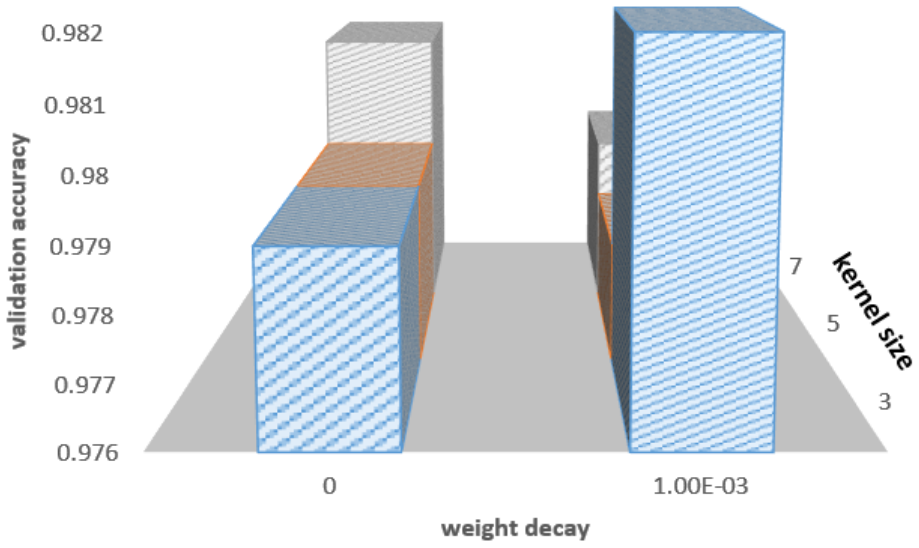

Figure 13: Comparison of validation accuracy with dropout.

By just looking at weight decay in isolation, it was difficult to determine what combination of kernel size, drop out, and weight decay would yield the most accurate model. In later stages of development, we realized that the most impactful changes did not involve optimizing LeNet hyper-parameters. Rather, data augmentation, network architecture as well as averaging models yielded significantly better results. We decided to forgo weight decay and dropout in the later models that we trained due to their insignificance towards model accuracy.

kernel size	weight decay	momentum	dropout(Y/N)	data augmentation	epoch	transfer learning	validation accuracy
5	0	0.5	N	None	30		0.978
5	1.00E-03	0.5	N	None	30		0.979
5	1.00E-04	0.5	N	None	30		0.977
5	1.00E-05	0.5	N	None	30		0.978
5	0	0.5	Y	None	30		0.977
5	1.00E-03	0.5	Y	None	30		0.978
5	1.00E-04	0.5	Y	None	30		0.978
5	1.00E-05	0.5	Y	None	30		0.977
5	0	0.9	N	None	30	Transfer MNIST	0.755
5	0	0.5	N	Normalise	30		0.978
5	0	0.5	N	Rotate	30		0.981
5	0	0.5	N	Random	30		0.985
5	0	0.5	N	Random	30	Last layer	0.981
5	0	0.5	N	Random	30	Complete retrain	0.987
5	1.00E-04	0.5	Y	Random	50		0.983
5	1.00E-03	0.5	Y	Random	50	Complete retrain	0.984
3	0	0.5	N	None	30		0.978
3	1.00E-03	0.5	N	None	30		0.977
3	0	0.5	Y	None	30		0.979
3	1.00E-03	0.5	Y	None	30		0.982
3	0	0.5	N	Random	30		0.985
3	1.00E-03	0.5	N	Random	30		0.985
3	1.00E-03	0.5	N	Random	30	Complete retrain	0.988
7	0	0.5	N	None	30		0.981
7	1.00E-03	0.5	N	None	30		0.981
7	0	0.5	Y	None	30		0.975
7	1.00E-03	0.5	Y	None	30		0.979

Table 3: LeNet models features vs. accuracy.

7 Discussion of results

7.1 Model Optimization

Looking at Table 3 , the highest validation accuracy models (98.4, 98.7, 98.8 % were all trained with random augmentations and retrained with the original KMNIST data set. Extending the number of epochs also slightly improved the validation accuracy of the models. Dropout was not conclusively beneficial towards LeNet's validation accuracy and there was no ideal combination of weight decay and kernel size.

Although our final submission models are not included in Table 3, we were able to achieve a final validation accuracy of 100 % through the extended LeNet5 architecture with a `RandomChoice` transform and retraining.

7.2 Ensemble Method: Model Averaging

Various models were combined using a majority vote method with the aim of producing a model with higher validation and test set accuracy. Combining five predictions from variations of trained LeNet5 and AlexNet models demonstrated promising improvements in the test accuracy when our team's submission test score rose from 96.1 to 97.7 % (by averaging predictions from models that had accuracies between 95.8 to 96.1 %). Later on, the predictions from this averaged model were combined with two variations of the extended LeNet5 models which

had test accuracies around 97.7 % . However, the resulting combined predictions did not improve the accuracy by as much as expected, demonstrating diminishing returns. The small increase in the test accuracy was interpreted as a consequence of the two extended LeNet5 models being highly interrelated and having misclassifications on the same images. Ideally, model averaging is used to combine a variety of different classifiers with different misclassifications. Therefore, averaging their predictions would correct some of the classification mistakes (i.e. the ones that are uncommon between the models) resulting in an overall improvement in accuracy.

If time permitted, training more LeNet and AlexNet models with different data-augmentation transformations could have resulted in less interrelated models and better model combinations.

8 Conclusion

From the results of this study, we have determined that the best performing LeNet models on classifying the KMNIST data set will have the following features:

- trained on an enlarged, augmented data set and retrained on the original data set
- trained through a greater number of epochs (at least more than 10)
- additional fully connected layers will improve the ability of the model to capture more information

Due to time constraints, we did not investigate techniques such as batch normalization, early stopping that may improve model accuracy. Further research towards improving the AlexNet architecture and more data augmentation may have also yielded greater results.

Finally, the model averaging method has shown great promise in delivering model improvements with insignificant computational overhead. Though the method demonstrated diminishing returns, we believe that given a larger number of models (and thus more possible permutations/combinations), we can further improve the accuracy of combined models.

References

[1] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based Learning Applied to Document Recognition.*Proceedings of the IEEE*. 1998;86(11):2278-2324.Available from: doi:10.1109/5.726791

[2] Alex K, Ilya S, Geoffrey E. H. ImageNet Classification with Deep Convolutional Neural Networks.*Communications of the ACM*. 2017;60(6):84-90.Available from: doi:10.1145/3065386

[3] Tarin C, Mikel B, Asanobu K, Alex L, Kazuaki Y, David H, Deep Learning for Classical Japanese Literature.*CoRR*. 2018; Available from: abs/1812.01718