CSE 472: Offline 4 Report

CNN from scratch

Najibul Haque Sarker
1705044

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

February 9, 2023

Contents

1 Introduction

The task is to build a Convolutional Neural Network from scratch without using any kind of deep learning framework. The training, validation and testing is done on the **numta**[1] dataset, which is a dataset containing hand written bengali digits with corresponding labels. The dataset contains 5 training-sets, of which Training-a, b and c are used for training and validation and training-d is used for independent testing. Training-e is excluded due to the very different type of data present there compared to the others.

2 Dataset and Training Split

Here Training a, b, c and d contains 19702, 359, 24298 and 10908 images respectively. So for training and validation, a total of 44359 images (a+b+c) are used and independent testing is done on 10908 images. Training a, b, c contained further metadata including the ***original database name*** and ***contributing team*** which made the dataset. As each training folder had a single contributing team, only the ***original database name*** attribute and the ***digit*** label is used to generate a stratified training and validation split where it is ensured that the data proportions of the unique values of these two metadata remain the same. This ensured the construction of a fair balanced validation set. The training and validation split is 80%-20%, meaning 35477 training and 8882 validation images are used.

During initial experimentation, a subset of the training and validation split is used due to the architecture chosen and resources limitation. The python package *imagededup* utilized CNN (mobilenet) embeddings to score duplicate images. Based on these scores, 13415 images are subsampled to form a 10721 training and 2694 validation set.

3 Experiments

The initial experiments was done on Model1 architecture with 64x64 shaped images (all the model descriptions are provided in Subsection 3.1). Due to resources constraint, all of the experiments done on this model was from the sub-sampled dataset containing 13415 images. Afterwards, LeNet [2] with 28x28 images was used using the full dataset i.e. 44359 images. Subsequently, 2 additional models Model2 and Model3 were used with 48x48 images. Of these experiments, Model3 is the final submitted model. Accuracy and Macro-F1 are the two metrices used and among them Macro-F1 is used for best model selection. Most of the experiments are done in Kaggle CPU notebooks.

3.1 Architectures

The details of the 4 model architectures along with the input sizes are provided in Table 1. Model1 are used for the initial experiments. Due to the resource hungry nature of the model, subsequent experiments are done using the LeNet model using 28x28 size images. But in order to capture more complex features using higher dimension images, Model2 is created by adding another extra Conv2D layer to the LeNet model. Model3 on the other hand is a variation of Model1 for 48x48 images and due to the lower image dimension, the resources needed for training became manageable. This is the model that is used for the final submission.

3.2 Data Augmentation

The provided dataset is mainly white background with the digit written in black. Experiments showed that providing the raw images gave low Macro F1 scores. Just inverting the image, meaning the white background becomes black and the black becomes white, increases the F1 score drastically.

Erosion, Dilation and Opening are very famous morphological operations used in computer vision. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. Opening operation is basically a sequence of erosion and dilation operations which removes the inherent noise of an image. Using only the dilation operation on the inversed image improves the score a lot, this amplifies the white written digit against the black background making detection of features easier.

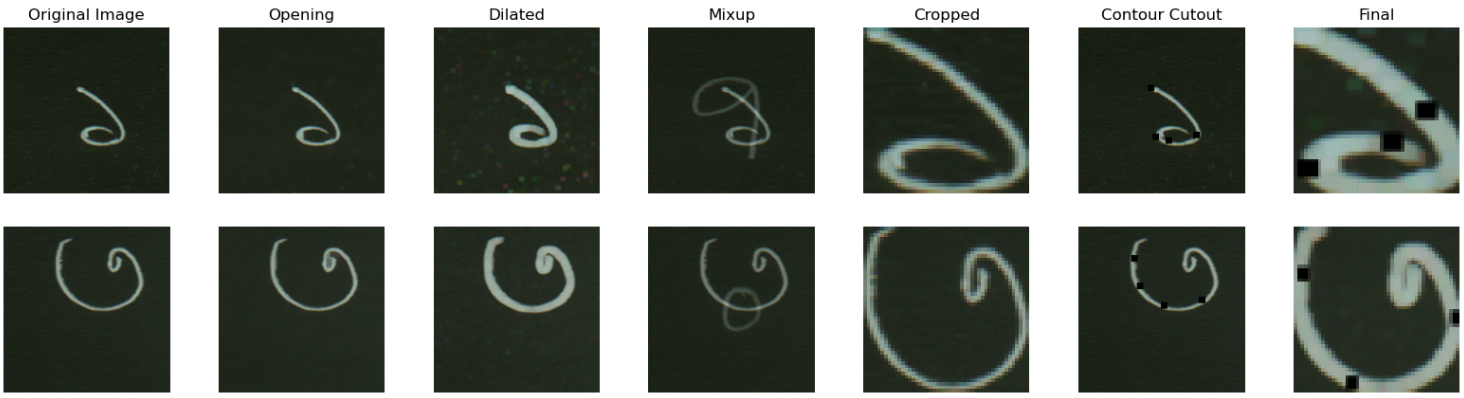


Figure 1: Different types of augmentations tested.

Mixup [3] is a famous augmentation technique which generated a weighted combination of random image pairs from the training data. The label for the corresponding image is similarly weighted. Experiments showed mixup may have a slightly adverse effect in the training of this dataset.

The images contained a lot of useless background and one idea is to remove this unnecessary background and only crop to the digit present. The opencv library is used to detect image contours after using opening operation to reduce noise. The image is

Model1	**LeNet**	**Model2**	**Model3**
• Input 64x64 • Conv2D (3, 16, 3, 1, 1) • ReLU • MaxPool2D (2, 2) • Conv2D (16, 32, 3, 1, 1) • ReLU • MaxPool2D (2, 2) • Conv2D (32, 32, 3, 1, 1) • ReLU • MaxPool2D (2, 2) • Flatten • Linear (2048, 1024) • ReLU • Linear (1024, 512) • ReLU • Linear (512, 10) • Softmax	• Input 28x28 • Conv2D (3, 6, 5, 1, 0) • ReLU • MaxPool2D (2, 2) • Conv2D (6, 16, 5, 1, 0) • ReLU • MaxPool2D (2, 2) • Flatten • Linear (256, 120) • ReLU • Linear (120, 84) • ReLU • Linear (84, 10) • Softmax	• Input 48x48 • Conv2D (3, 6, 5, 1, 0) • ReLU • MaxPool2D (2, 2) • Conv2D (6, 16, 5, 1, 0) • ReLU • MaxPool2D (2, 2) • Conv2D (16, 32, 3, 1, 1) • ReLU • MaxPool2D (2, 2) • Flatten • Linear (512, 120) • ReLU • Linear (120, 84) • ReLU • Linear (84, 10) • Softmax	• Input 48x48 • Conv2D (3, 16, 3, 1, 1) • ReLU • MaxPool2D (2, 2) • Conv2D (16, 32, 3, 1, 1) • ReLU • MaxPool2D (2, 2) • Conv2D (32, 32, 3, 1, 1) • ReLU • MaxPool2D (2, 2) • Flatten • Linear (1152, 1024) • ReLU • Linear (1024, 512) • ReLU • Linear (512, 10) • Softmax

Table 1: Overview of the 4 architectures. Here are the parameter details: Conv2D(in channels, out channels, kernel size, stride, padding), MaxPool2D(kernel size, stride), Linear(in features, out features).

then cropped to the bounding box of the contours to extract the region of interest. This augmentation greatly improved the score.

Sometimes when writing, some parts of the digit gets thinned out or attenuated due to defects of the pen or writing material. To address such cases, a new augmentation is introduced named Contour Cutout. Basically this is a modification of the famous Cutout [4] algorithm, where instead of randomly cutting squares from the image, the squares are instead cut from random points along the digit. This is done via contour detection and shows some improvement over the baseline.
All of these augmentations are used in the final model except opening and mixup.

3.3 LR Scheduler

The learning rate has a significant effect on the training of a model. From experiments, it came to light that a lr of 0.01 was the best learning rate for this experiment when training from scratch. But keeping the learning rate constant throughout the training can have adverse effect as the model might try to explore more even when finding a global minima/maxima. In order to tackle this, a variant of ReduceLROnPlateau scheduler is implemented which lowers the learning rate by a predetermined factor when the validation MacroF1 score doesn't increase after some predetermined number of epochs.

3.4 Pretraining

Pretraining on a large set of dataset and then applying transfer learning makes the model learn more easily. Keeping this in mind, the EMNist or Extended-MNist dataset [5] is used to pretrain the models. This dataset consisted of 240,000 training images and 40,000 testing images. The pretraining of the models didn't increase the validation score, but it made the models reach their optimized points quicker. Due to the resource constraints, this proved useful during tuning.

3.5 Pseudo Labels

The training dataset also included unlabelled datasets of testing-a, b, c and d. Of these, pseudo labels were generated for testing-a, b and c using an ensemble of trained teacher models. Then a student model is trained in a supervised fashion with labeled and unlabeled data simultaneously. Pseudo labels has an effect similar to entropy regularization [6], and has the potential to make the model more robust. Here, pseudo labelling slightly increased the score for the trained models. Just to clarify, **no** images from the independent trainig-d was just during this experiment.

Model	Aug	lr	Extra	Val Macro F1
Model1 (64x64)	N/A	0.001		0.1549
	Reverse			0.48
	BBox, Reverse			0.8859
	BBox, Reverse, Dilation			0.8929
	BBox, Reverse, Mixup			0.8799
LeNet (28x28)	Reverse	0.01		0.8858
	Reverse, Dilation	0.01		0.8837
		0.001		0.7353
	BBox, Reverse, Dilation	0.01		0.9598
		0.001		0.9202
Model2 (48x48)	Reverse, Dilation	0.001		0.7657
	BBox, Reverse, Dilation	0.001		0.9206
		0.01		0.9662
	BBox, Reverse, Dilation, Contour Cutout	0.01		0.966
		0.01	Pseudo	0.9704
		0.01	Pretrained, Pseudo	0.9711
Model3 (48x48)	Reverse, Dilation	0.001		0.7285
	BBox, Reverse, Dilation	0.001		0.9378
		0.01		0.9643
	BBox, Reverse, Dilation, Contour Cutout	0.01	Pseudo	0.9769

Table 2: Ablation Study

4 Results

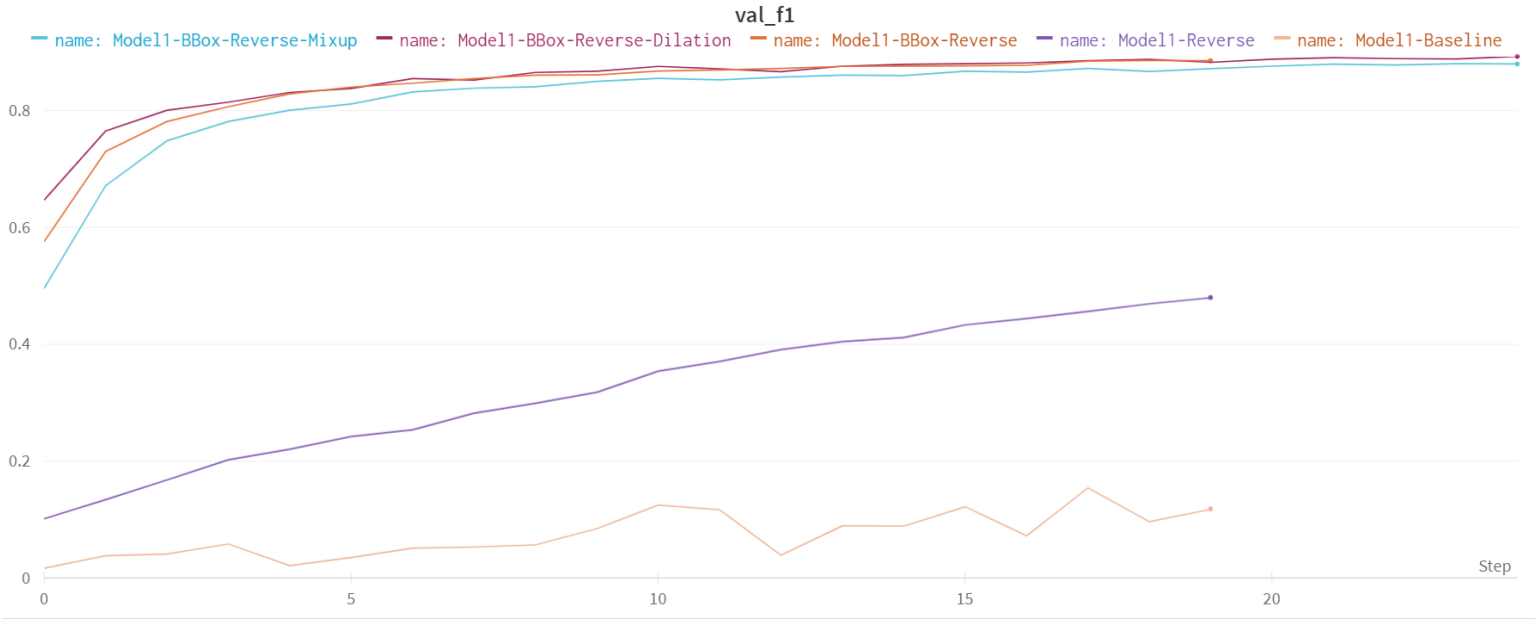
The whole ablation study is given in Table 2.

4.1 Model1

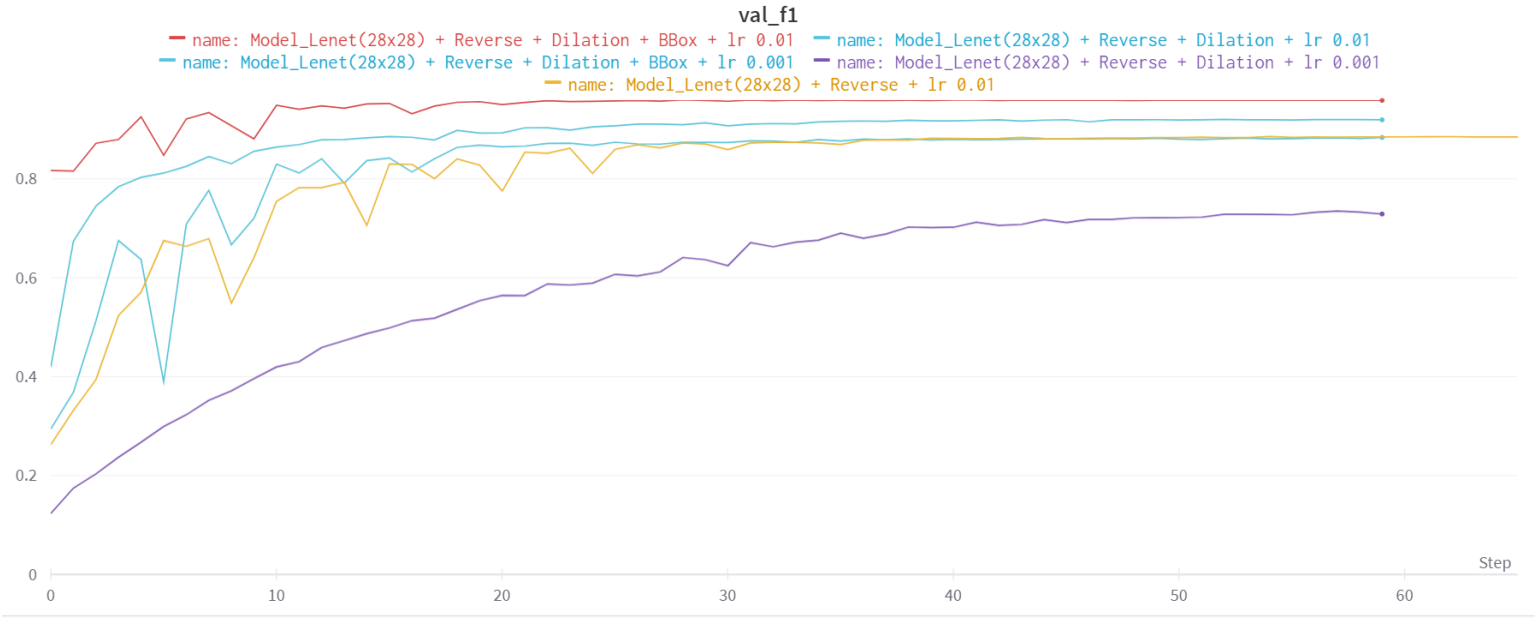
This experiment was done on the subsampled dataset with image size 64x64. This made it clear that using reverse and cropped bbox augmentation gave a huge boost to the model. The validation F1 curves are given in Fig. 2b.

4.2 LeNet

Due to resources constraint, experiments were switched over to 28x28 images and LeNet was the perfect architecture for this. These experiments made it apparent that lower dimension images on such architectures performed better on this dataset. Furthermore, this revealed that lr 0.01 performed better than lr 0.001 and this also necessitated the use of the lr scheduler. The validation F1 curves are given in Fig. 2b



(a) Model1 Experiments



(b) LeNet Experiments

4.3 Model2

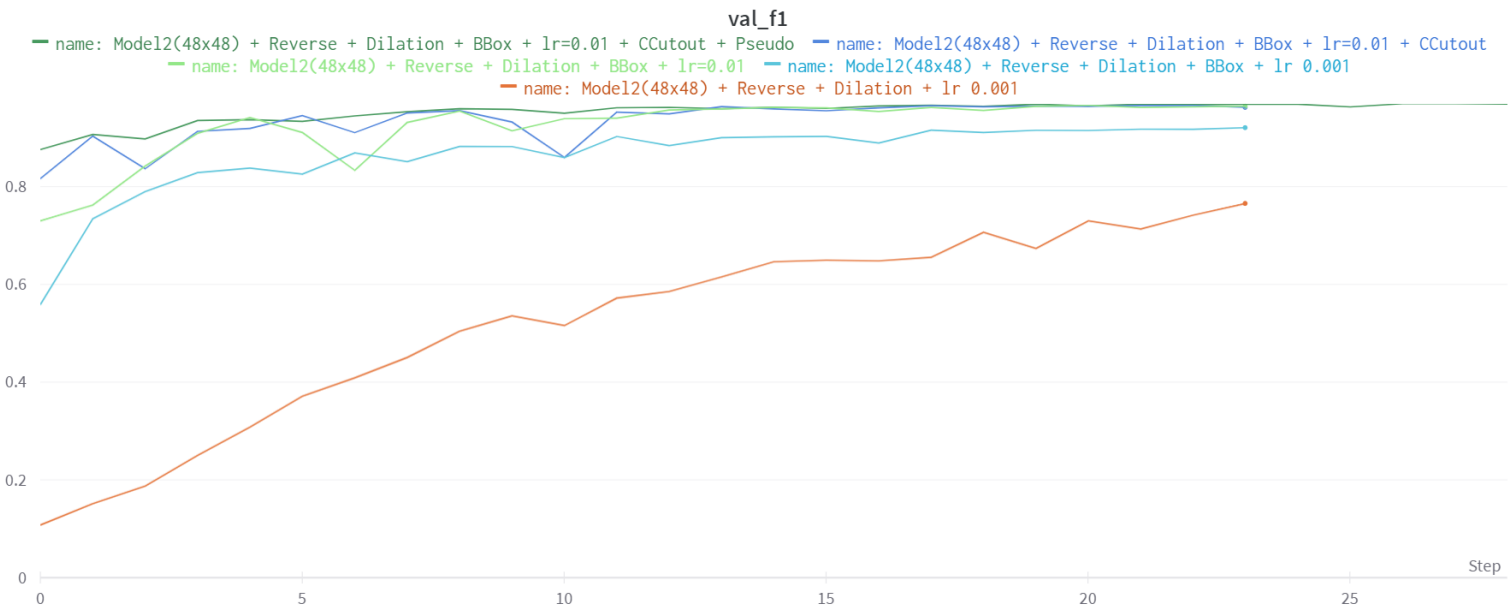


Figure 3: Model2 Experiments

The test score of Model LeNet hinted towards the model not being diverse enough or the image dimension big enough. That is why Model2 was created to train on 48x48 sized images. The results shown in Fig 3 make it apparent that 48x48 images indeed can provide more rich features and make a robust model. These experiments also provided evidence that pseudo labelling can increase the models performance on the independent test set.

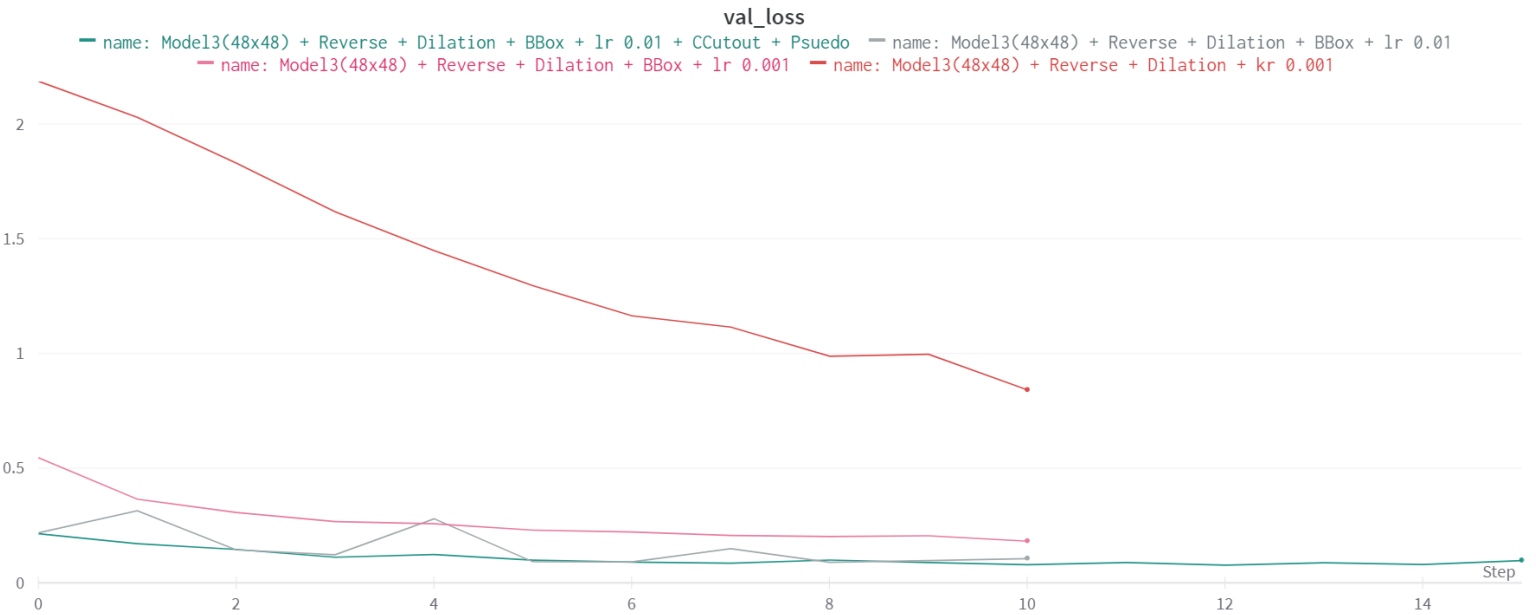
4.4 Model3

Following this idea, the Model1 was further adapted to include 48x48 images. This also proved fruitful due to the validation macro f1 and accuracy shown in Fig 4b and training and validation loss shown in 5b. This model has more filters in Convolution layers and has more features in fully connected layers than Model2. Thus the knowledge capacity is greater but that makes it harder to train too.

(a) Model3 validation Macro F1

(b) Model3 validation Accuracy

(a) Model3 training loss



(b) Model3 validation loss

Model	Validation Macro F1	Test Macro F1
Model2	0.966	0.97506
Model2 (pretrained + pseudo)	0.9711	0.98139
Model3 (pseudo)	0.9769	0.98133

Table 3: Final Models Comparison

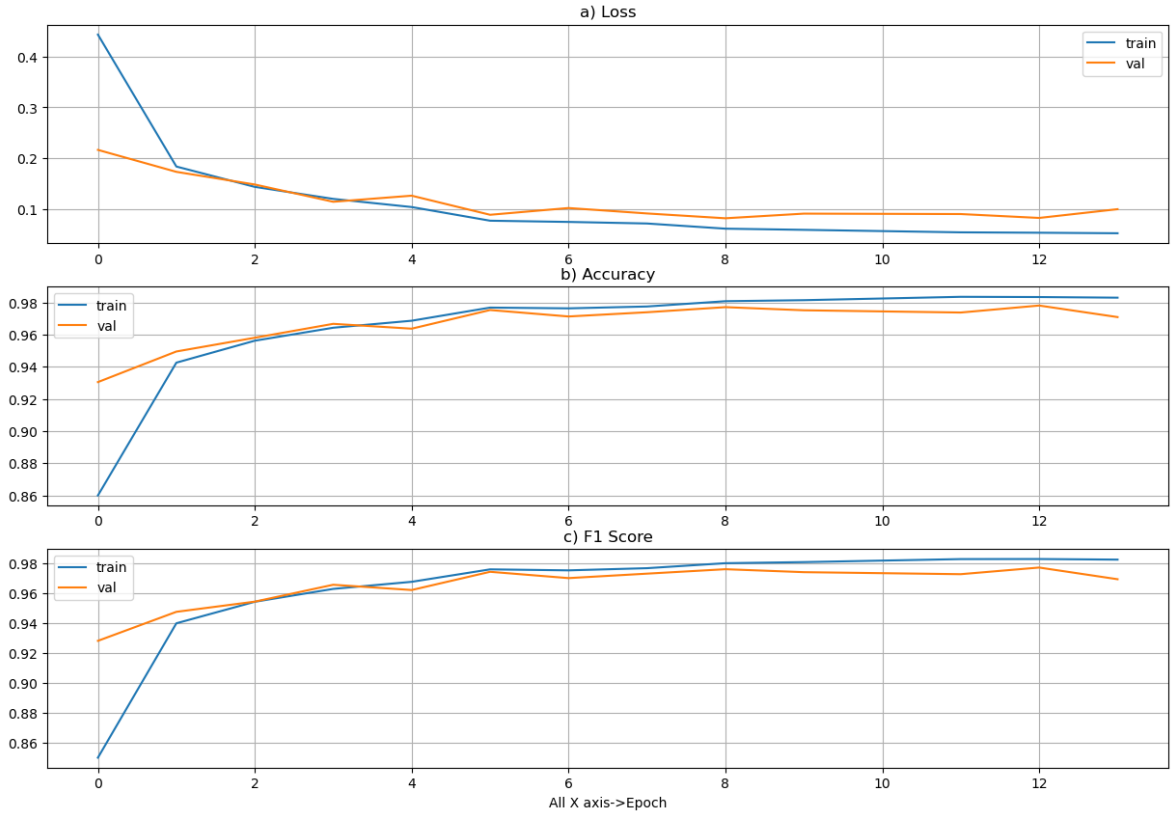
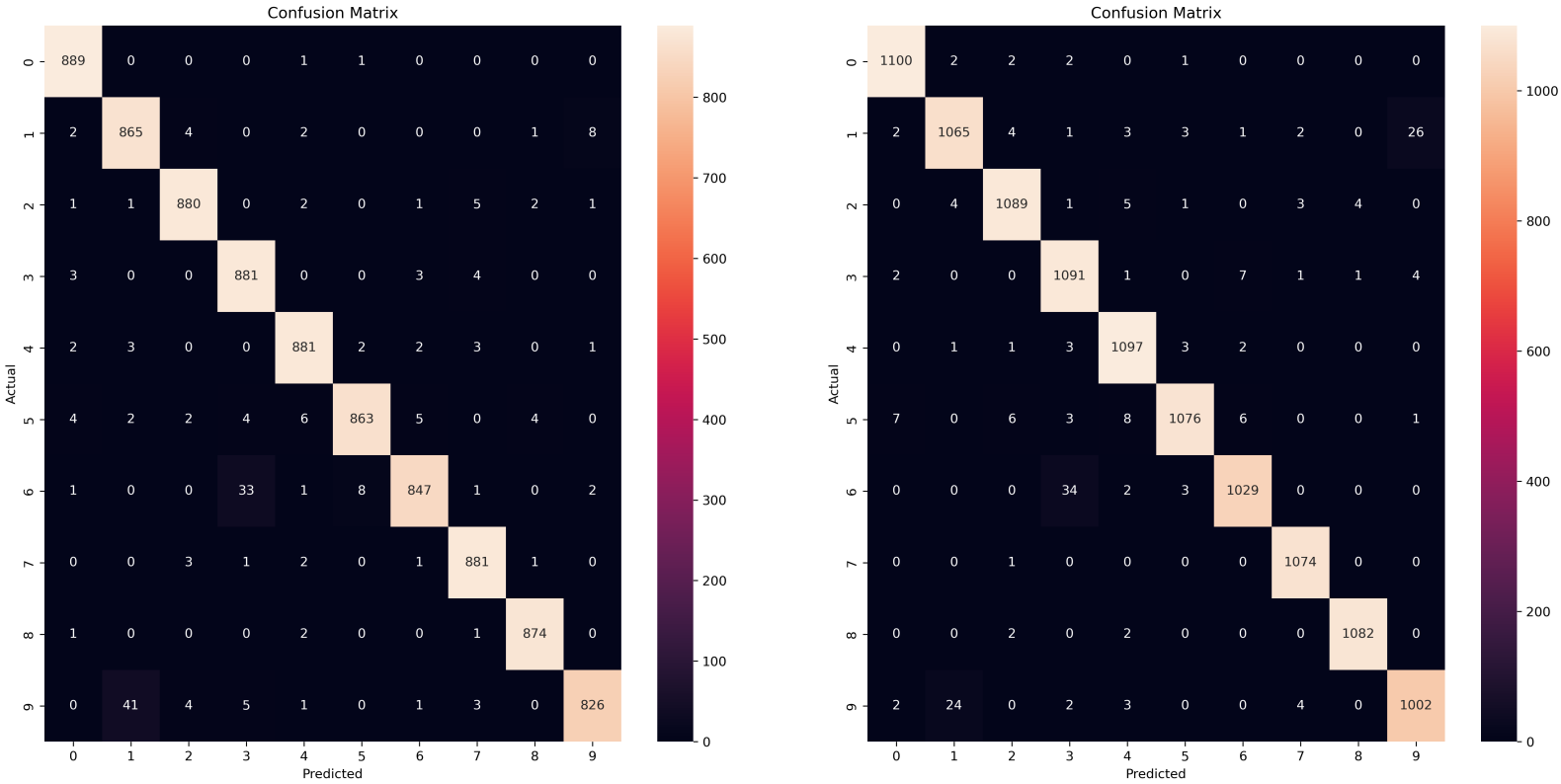


Figure 6: Final Model Scores

5 Final Model

The final model is chosen from the topmost validation Macro F1 scores. Their comparison with the test Macro F1 is provided in Table 3. Here all the models have BBox, Reverse, Dilation, Contour Cutout for augmentation and initial learning rate of 0.01 with an Reduce-lr-on-plateau scheduler.

Thus, Model3 (pseudo) is the final model that is chosen. This had the most validation Macro F1 of 0.9769 and its test Macro F1 score is 0.98133. The training vs validation loss, Macro F1 and accuracy graphs are given in Fig 6 and confusion matrixes are given in Fig 7. All the logs are available here at: wandb logs.



(a) Validation

(b) Test

Figure 7: **Confusion Matrixes of Validation and Test**

Running the script

In order to run the train script, there must be a config.yaml file which has all the configurations set there (please look at the example config given). The script has wandb logging support, to use this you must have an account in wandb and update the related details in the config.yaml.

To run the inference script, you need to provide the testing directory in the command line argument. If the *gt_csv* attribute in the config.yaml is not set to *false* and set to a csv file path, then the script will also compute macro_f1, accuracy and confusion metrics of the inferred result against the csv file contents. The config.yaml must have the *checkpoint_path* attribute set to the model's weights path to load that model for inference. The model architecture must also be provided.

References

[1] Samiul Alam et al. *NumtaDB - Assembled Bengali Handwritten Digits*. 2018. DOI: 10.48550/ARXIV.1806.02452. URL: https://arxiv.org/abs/1806.02452.

[2] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[3] Hongyi Zhang et al. *mixup: Beyond Empirical Risk Minimization*. 2017. DOI: 10.48550/ARXIV.1710.09412. URL: https://arxiv.org/abs/1710.09412.

[4] Terrance DeVries and Graham W. Taylor. *Improved Regularization of Convolutional Neural Networks with Cutout*. 2017. DOI: 10.48550/ARXIV.1708.04552. URL: https://arxiv.org/abs/1708.04552.

[5] Gregory Cohen et al. *EMNIST: an extension of MNIST to handwritten letters*. 2017. DOI: 10.48550/ARXIV.1702.05373. URL: https://arxiv.org/abs/1702.05373.

[6] Dong-Hyun Lee. "Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks". In: *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)* (July 2013).