no-reply@qut.edu.au
To: Rodo Nguyen
Sun 15/05/2022 22

QUT | HiQ

Hi Dac Duy Anh,

Thank you for your assignment extension request **(FORM-AEX-90722)**.

We have approved your request and the due date for your assignment **Assignment 1B**, for unit CAB420 has been extended by 48 hours from the original due date. If your unit outline does not specify that your assignment is eligible for an extension, this confirmation email is not valid and unless you submit by the original due date, the late assessment policy will apply.

A copy of this email must be attached to the front page of your assignment when you submit it.

You are responsible for:

• Ensuring that this assignment is eligible for extension before submitting it after the original due date.

• Attaching a copy of this email to your assignment when you submit it.

Be aware that a copy of this email is kept on file. You should not alter this email in any way. Email notifications that have been altered or differ in any way from the original may result in an allegation of student misconduct as set out in the Student Code of Conduct.

Need extra support? You can access free, confidential counselling with qualified professionals. We also offer planning and support if you have a disability, injury or health condition that affects your ability to study. If you are struggling to meet your university commitments, academic support is also available.

Have a question? Check our online help on late assessment and extensions. We're also available to help you, visit the HiQ website for contact details and opening hours.

Email us | +61 7 3138 2000

HiQ is your place to go for **student services, support** and **general enquiries:** qut.to/abouthiq

ASSIGNMENT 1B

CAB420 Machine Learning, Semester 1 2022

Student: Dac Duy Anh Nguyen (Rodo Nguyen)

Student ID: 10603280

Table of Contents

Problem 1: Training and Adapting Deep Networks

1 Data Pre-processing

The given dataset is colour and in the size of 32x32. The images show the number clearly and some images contain more than one number (Appendixes 1).

For SVM method, no validation data is required for tuning hyper-parameters so the whole train data is used for training. For deep networks, validation is randomly got from the 10% of our original 1000 train data points. This gives the models a major part of the samples to train on while all classes are ensured to present in the validation data.

In this problem, the dataset is converted to gray scale since the gray ones still keeps the important features (e.g. edges, shapes, lines, etc.) to identify numbers while colour is deemed to play a minor role. This also helps to reduce the computational cost for training our Deep Networks. The image size is retained the same as it is the suitable to contain enough details so that the model is able to learn but at the same time it is not too large to increase the number of model parameters unncessarily. The classes (y values) are left as numerical values since they are small values (from 0 to 9) and thus do not affect the learning process.

It is worth noting that the data distribution is not equal between classes (Appendixes 2). However, the bias impact is not significant in prediction results from Deep Network models but it impacts heavily to the SVM – a non-deep model. Further discussion is presented in section **3 Model Training**.

2 Neural Network Model Design

In this project, the VGG-style network (VGG) is chosen over Residual Network (ResNet) as ResNet is more suitable for building deep model to solve complex problems (Mujtaba, 2020) and classifying numbers is not considered to be one of them. Furthermore, as model gets deeper, we need a large amount of data so that the layers in the end have more opportunities to learn and adjust their parameters. The given dataset in this problem contains only 1000 samples for 10 classes so a deep model using ResNet or even VGG is not necessary and will only make the learning process harder and the benefits of accuracy increase may not outweigh the cost of computing time on our available hardware (see Appendixes 3) and power.

With the reasons discussed above, our VGG model used in this problem only consists of 3 Convolutional blocks and takes input of shape 32x32x1 (gray 32x32 image). Each Convolutional block has 2 Convolutional layers with 3x3 kernal, 1 Batch Normalization layer to normalize the output of a batch and facilitate the learning process of later layers, 1 Rectified Linear Unit (ReLU) activation layer to only allow useful weight pass through, and 1 2x2 Max Pool layer to reduce the input dimentionality while keeping the most significant features.

With 3 blocks of convolutions, 3x3 kernal is able to capture information local information in the beginning and larger, more general features in the later block.

The final Convolutional block does not have Max Pool layer but a Flatten layer to turn the input into a vector. 3 Dense layers follows to learn important information and designed to give 10 outputs according to 10 classes we are to classify. Except for the first 2 Dense layers using ReLU, the final one uses Soft Max to highlight and select the highest class while surpressing others. Dropout layer is inserted to reduce overfitting. But it may also cause slow learning so plenty of epochs is given (until no improvement is observed) to overcome this.

Besides that, Sparse Categorical Crossentropy is utilized as a loss function since our output is left as labels and it is suitable for multi-classification problem to measure the inaccuracy between labels and predictions. As SoftMax is used in our model, from_logits parameter of this loss function is set to False.

Adam (Adaptive Moment Estimation) optimizer is chosen due to its memory efficiency and the ability to adaptively adjust its learning rate depending on each learning stage and its momentum to escape local minima and reach global minimum (Brownlee, 2017).

Layer (type)	Output Shape	Param #
img (InputLayer)	[(None, 32, 32, 1)]	0
conv2d_24 (Conv2D)	(None, 32, 32, 8)	80
conv2d_25 (Conv2D)	(None, 32, 32, 8)	584
batch_normalization_12 (Batc	(None, 32, 32, 8)	32
activation_12 (Activation)	(None, 32, 32, 8)	0
spatial_dropout2d_12 (Spatia	(None, 32, 32, 8)	0
max_pooling2d_8 (MaxPooling2	(None, 16, 16, 8)	0
conv2d_26 (Conv2D)	(None, 16, 16, 16)	1168
conv2d_27 (Conv2D)	(None, 16, 16, 16)	2320
batch_normalization_13 (Batc	(None, 16, 16, 16)	64
activation_13 (Activation)	(None, 16, 16, 16)	0
spatial_dropout2d_13 (Spatia	(None, 16, 16, 16)	0
max_pooling2d_9 (MaxPooling2	(None, 8, 8, 16)	0
conv2d_28 (Conv2D)	(None, 8, 8, 32)	4640
conv2d_29 (Conv2D)	(None, 8, 8, 32)	9248
batch_normalization_14 (Batc	(None, 8, 8, 32)	128
activation_14 (Activation)	(None, 8, 8, 32)	0
spatial_dropout2d_14 (Spatia	(None, 8, 8, 32)	0
flatten_4 (Flatten)	(None, 2048)	0
dense_12 (Dense)	(None, 256)	524544
dropout_4 (Dropout)	(None, 256)	0
dense_13 (Dense)	(None, 64)	16448
dense_14 (Dense)	(None, 10)	650

Total params: 559,906
Trainable params: 559,794
Non-trainable params: 112

Figure 1 The VGG model used in this Problem

3 Model Training

3.1 SVM one-vs-one

For non-deep classifier, SVM One-vs-One (will be mentioned as SVM onwards in this report for short) is implemented with C=1 and other hyper-parameters are left to default settings.

3.2 VGG: Training from scratch (Model 1)

The model is trained with batch size of 32 and 200 epochs max. The training utilized 2 callbacks: ModelCheckpoint to only save the last model that has improvement in validation accuracy and EarlyStoping to stop the training if no improvement in validation accuracy is seen after 10 epochs. Analysis shows that whether to choose model with the best validation loss or accuracy have little impact on the final performance and sometimes they are the same model.

A suitable batch size were also considered and tested immensely. Too small batch size (less than 10) will not allow the model to learn all the possible classes and too large batch size will not allow the model many steps to update its parameters. For instance, train data of 1000 samples with 128 images in a batch will only allow the model updates 8 times per epoch. Further experiments shows that 16 to 64 samples per batch produces similar results and thus are used in the VGG models.

In addition, the same 2 callbacks' setting is used in the remaining VGG models.

3.3 VGG: with Augmentation Layer (Model 2)

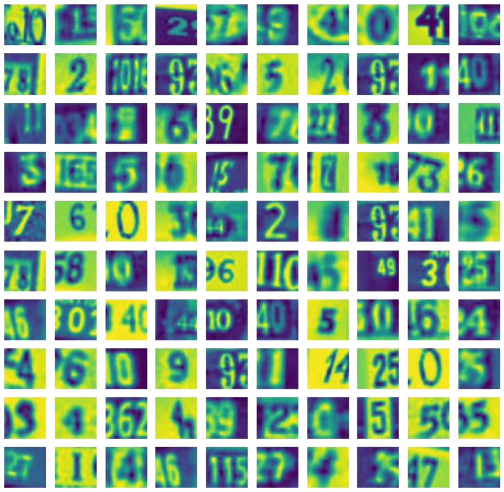


FIGURE 2 AUGMENTED SAMPLES

An augmentation layer is added and applied after the Input layer consists of changes as below:

- Random rotation: 0.05
- Random zoom: height_factor=(0, 0.1), width_factor=(0, 0.1)
- Random translation: height_factor=(-0.025, 0.025), width_factor=(-0.025, 0.025)
- Random contrast: 0.2

An inspection of the output images suggests that all the augmented images keep their original meaning. Some are shown in Figure 2.

Batch size of 32 is used in this model.

3.4 VGG: with Fine-tuning (Model 3)

For the fine-tune task, the pre-trained model – 'vgg_3stage_MNIST_small' – from CAB420 resources is selected and adapted to this specific problem. The model is trained on MNIST dataset and has learned to recognise number features and classify numbers, which is very similar to this problem.

Importantly, the original model design finds it difficult to learn the new data and no improvement is shown in training. An initial speculation is that many Dropout and Batch Normalisation layers in the end caused the much lost information. Therefore, the last 9 layers of this model are replaced with the last 4 layers from our designed VGG models (see Appendixes 4). This action brings the improvement in loss and accuracy as I wanted.

SGD optimizer is utilised due to its suitability for fine-tuning task.

However, our current dataset has to be resize down to (28,28,1) to fit into the selected model's Input layer. Every layer remains trainable to maximize the ability to adapt to the new data.

4 Comparison & Evaluation

General

Overall, it is clear that all the CNNs have performed much better than the SVM has as shown in Table 1 below. Training time of CNNs is also significantly higher. Nonetheless, training time is a one-time cost and can be neglected in practical circumstances where accuracy and fast prediction time are much more important. Additionally, the training time of SVM is linearly correlated with the amount of train data while CNN's is not (Appendixes 5). So in the scenarios where data is abundant and quality, CNNs will have more advantages as it get better and better with more train data and the inference time remains the same or only increase by a small margin or even reduced with more powerful hardware.

	SVM	Train from scratch	Train with Augmentation layer	Fine-tune
Training time (s)	0.74	302.38	127.91	96.17
Inference time (s)	11.50	12.40	12.07	9.48
No. Epochs used	na	63	97	88
Test accuracy	14.9%	78.17%	78.49%	80.57%

TABLE 1 RESULTS OF 4 IMPLEMENTED METHODS

SVM model

As for SVM mechanism, the SVM model only predicts new samples based on the separators it draw on hyperplanes of previous learned samples unlike the VGG or Convolutional Neural Network model in general, they learn and store various features of the train data in its parameters. Therefore, the SVM prediction result is heavily affected by the quality and quantity of the train data. In this case, the train data is not equally distributed, which lead to prediction bias in the major classes in both train and test stages.

As can be seen in Appendixes 2 regarding samples distribution, with many samples concentrate in class 1 and 2 considerably, lots of our SVM predictions fall falsely into these classes as well (Figure 3) as a result of imbalance dataset. This bad performance is also indicated in F1-score (Figure 3). They are all below 0.45 in the scale of 1 and average at 0.35 while class 1's is twice as great as many other minority classes, e.g. 0, 6, 7, 8, 9.

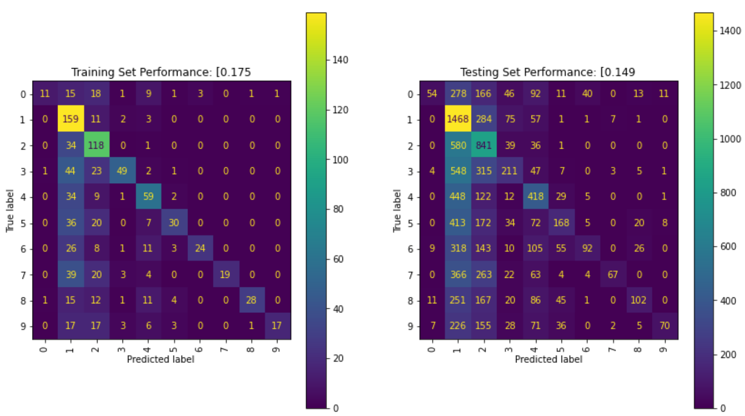


FIGURE 3 SVM MODEL: PREDICTION RESULT

	precision	recall	f1-score	support
0	0.64	0.08	0.14	711
1	0.30	0.78	0.43	1894
2	0.32	0.56	0.41	1497
3	0.42	0.18	0.26	1141
4	0.40	0.40	0.40	1035
5	0.47	0.19	0.27	892
6	0.62	0.12	0.20	758
7	0.85	0.08	0.15	789
8	0.59	0.15	0.24	683
9	0.77	0.12	0.20	600
accuracy			0.35	10000
macro avg	0.54	0.27	0.27	10000
weighted avg	0.48	0.35	0.30	10000

FIGURE 4 SVM MODEL: CLASSIFICATION REPORT

VGG models

Noticeably, model 1's loss and accuracy converge faster than the other models' due to the little train data and stops its training at epoch 63. In contrast, thanks to the Augmentation layer, model 2 contantly learns and improves from augmented images and only stops at epoch 97.

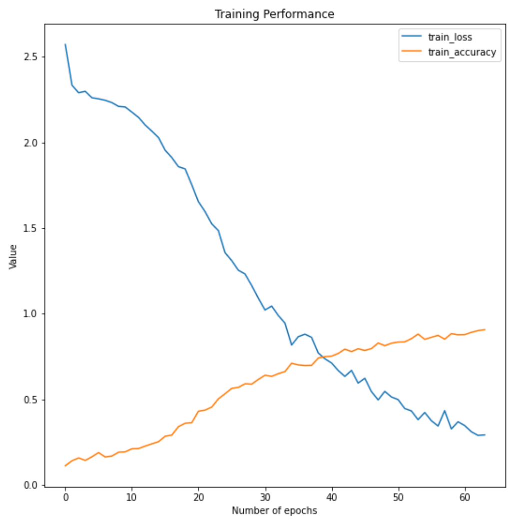
However, model 2 does not bring a significant improvement in test accuracy as we expected with an extra Augmentation layer, gaining a 0.5% increase compared to the model 1.

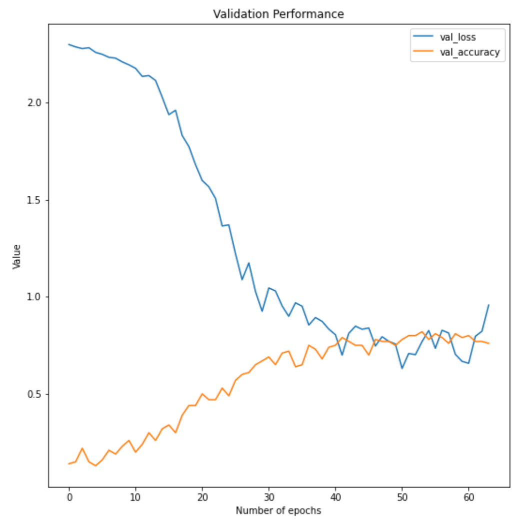
The pretrained model performs best with the highest test accuracy of 80.57%. use less epochs than Augmentation and even significantly less training time (aproximately a third) but gives us a good increase to 80.57% in test accuracy thanks to its pretrained parameters. It also learns slightly faster in the beginning compared to the other two models as shown by the steeper curves on both the loss and accuracy. Interestingly, the lowest validation loss in model 3 is still greater than ones in model 1 and 2's.

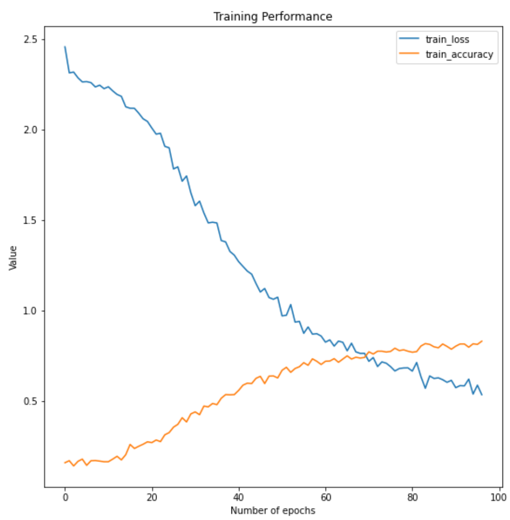
A problem that may have prevented model with Augmentation Layer and pre-trained model from being more accurate is the validation data. The validation data is imbalance and too little for a deep network to learn false prediction and backpropagate efficiently (i.e. adjust the model's parameters). The issue is also expressed by the big spikes and strong fluctuation in validation loss/accuracy (especially in model 3) as the models started to overfit.

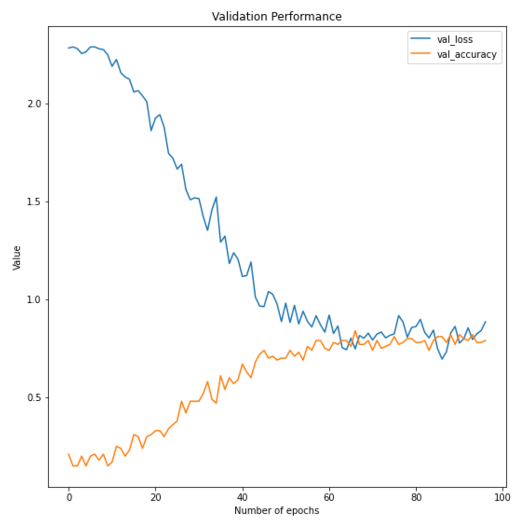
With that reason, we speculate that a bigger and balance train and validation dataset will improve the performance and reveal the difference between 3 models' performance substantially.

Looking at Classification report in Appendixes 6, the evidence of imbalance data's impact become clear as F1-score of dominent classes is higher than the other minors.









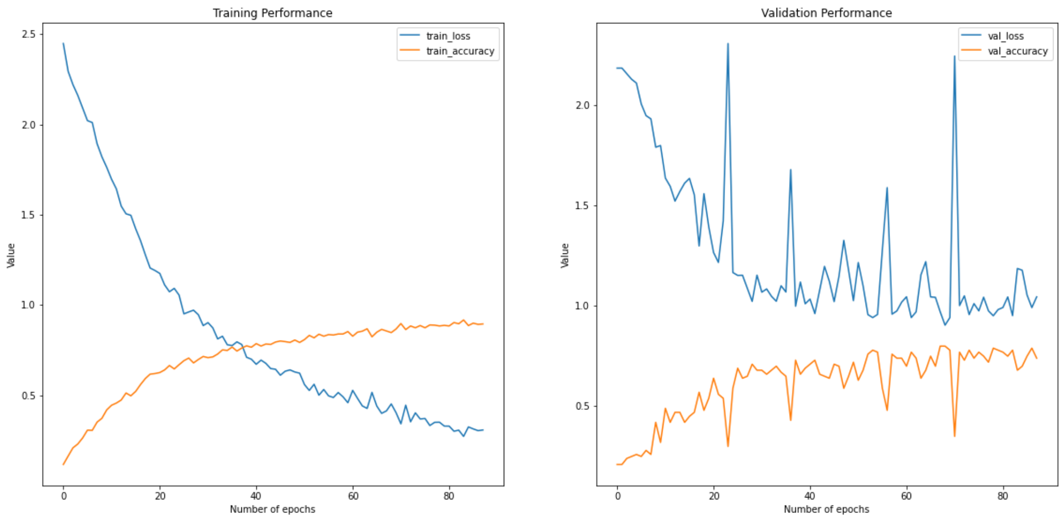


FIGURE 5 LOSS AND ACCURACY IN TRAINING 3 VGG MODELS

Problem 2: Person Re-Identification

1 Data Pre-processing

The images are resized from 128x64 to 64x32 just for deep learning based method to reduce the computational cost of training. 20% of the train data is randomly selected and used for validation purpose. However, it is worth noting that some classes in validation data have 0 sample as a consequence of highly imbalance dataset. This may negatively affect the learning and adjusting parameters process of the model as well.

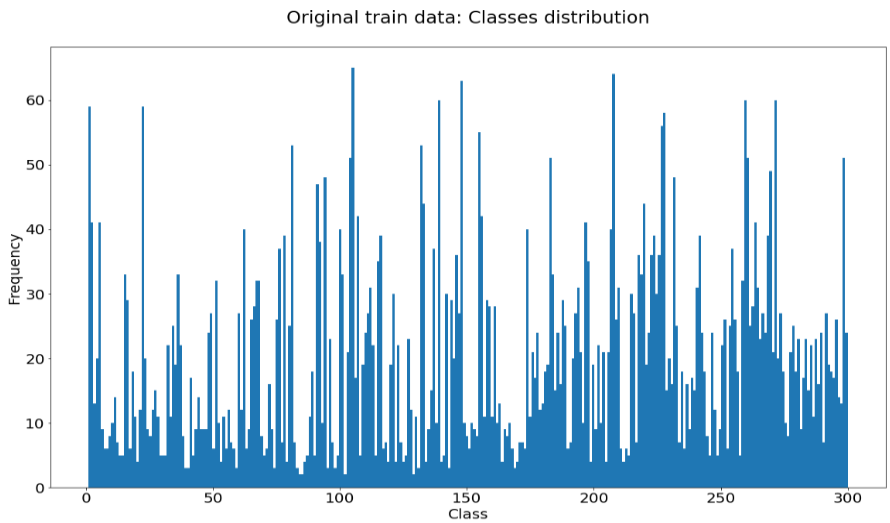


FIGURE 6 DATA DISTRIBUTION IN TRAIN DATASET

Besides, the test data is perfectly balance. However, it does not play any role in the training stage except for providing us a fair test accuracy.

The images are not converted to gray scale as colors of clothes are considered to be crucial in helping distinguish people since other features are very similar (e.g. human pose) and facial details are not clear in such low dimension images.

2 Match Calculation

Match statistics was calculated by first transforming gallery and probe images into vectors. Then, a total distance from every gallery to a probe is calculated. The type of distance was considered between L1, L2, and Consine. Experiments from several trained model shows that L2 and Consine distance produced similar performance and much better than L1 did (as indicated in a steeper Cumulative Match Characteristics or **CMC** curve). So in the implementation phase, L2 is selected for its simplicity.

2 Model Design & Training

2.1 Non-deep learning method

Principal Component Analysis (PCA) is implemented as a method to reduce dimensions of the dataset. LDA may not be the best method for this problem because of the limited samples available.

It is worth noting that the number of samples (5933) is smaller than the number of dimensions (64x32x3=6144). So possible instability performance in PCA was taken into account as well.

Cummulative sum of PCA explained variance is ploted from PCA fitted on vectorised train data as below. Since little computation is required in this non-deep method, dimensions is decreased from 5933 down to 2559 by selecting the 99.99% most explained variance. That is already a 67% decrease. An examination on the reconstruction proves that PCA works properly.

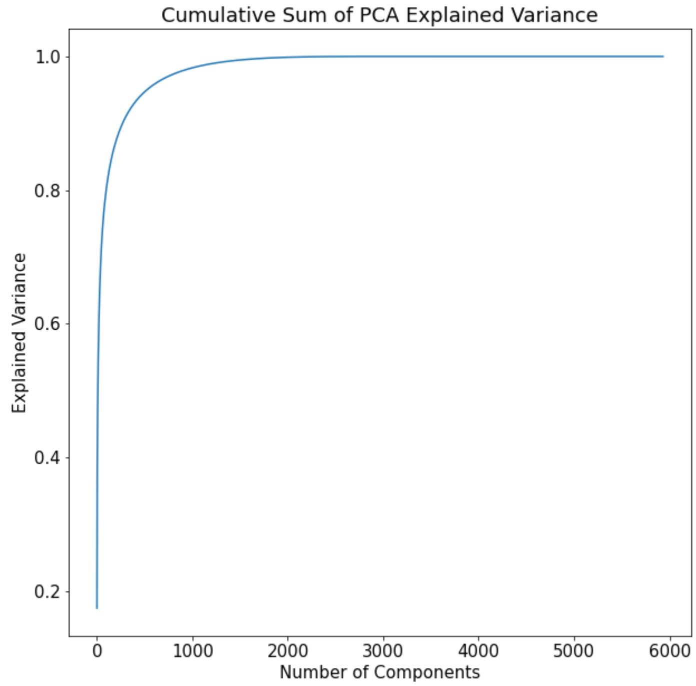


FIGURE 7 CUMULATIVE SUM OF PCA EXPLAINED VARIANCE

The result of Top-N accuracy calculated using 2559 components is as followed:

- Top-1: 10.63%
- Top-5: 16.61%
- Top-10: 25.58%

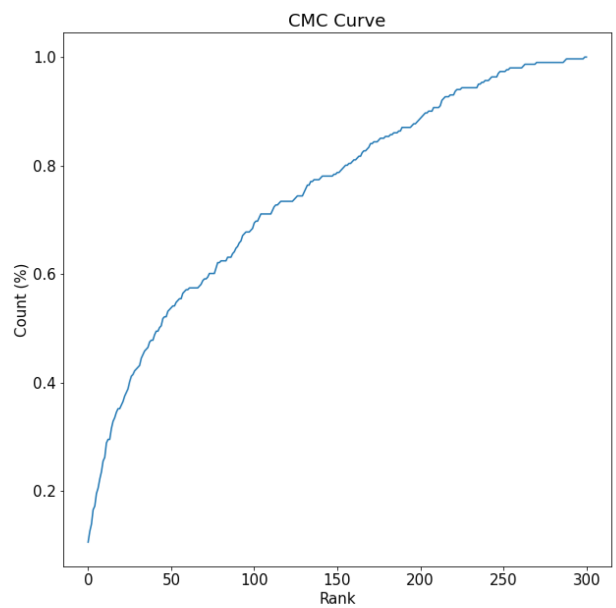


FIGURE 8

2.2 Deep learning based method

The Siamese model with triplet loss is implemented because of its strength by taking advantage of both negative and positive cases. The base network uses a VGG network which is simple but efficient in learning images. Again, ResNet was considered but may not be the best model due to the same reasons discussed in Problem 1: small number of samples/low-quality data, simple problem and hardware constraints.

The VGG network (see Appendixes 7) includes 3 blocks of (2 Convolutional, 1 Batch Normalisation, 1 ReLU activation, 1 Spatial Dropout and 1 Max Pooling layers). The last block replaces the Max Pool layer by a Fully Connected layer and is followed by a Dense, Batch Normalisation, ReLU activation and finally an embedding layer of size 32.

The head of base network is attached by 3 Inputs: Anchor, Positive and Negative. The end layer of this model is Triplet loss function using L2_normalize distance.

Batch size is set to 128 to both allow the model to learn various samples at a time and avoid data imbalance's impact while the plenty of epochs (max epochs: 100) is given to allow many opportunities to update its parameters. 2 callbacks are included: ModelCheckpoint to save only the model with lowest validation loss and EarlyStopping to stop the training when no loss decrease is detected in 5 epochs.

The result of Top-N accuracy are as below:

- Top-1: 25.25%
- Top-5: 50.50%
- Top-10: 64.12%

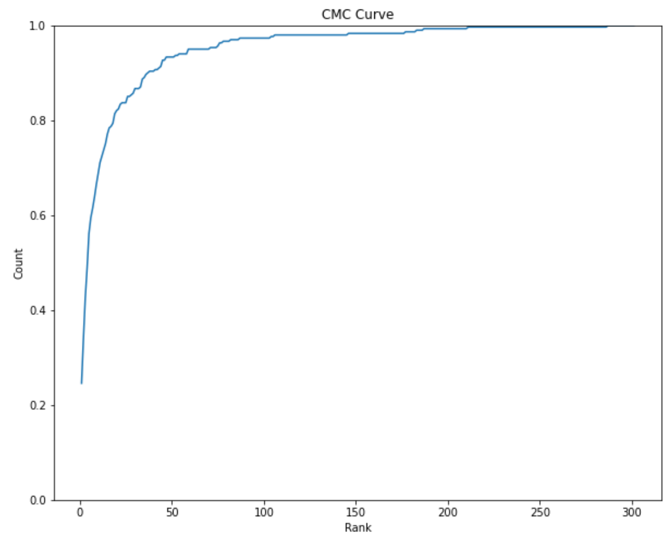


FIGURE 9

The CMC curve gives us a steep curve which indicates the network predicts relative good considering the defects of its dataset.

3 Comparison & Evaluation

With the results presented, the Siamese clearly performs much better than the non-deep method – PCA. A large part of it is because Siamese network is can learn and remember images features within its parameters while PCA merely change the structure of the train data and do not learn from it.

And the Siamese network has much potential to improve when we select and optimize the base network or use the original image size.

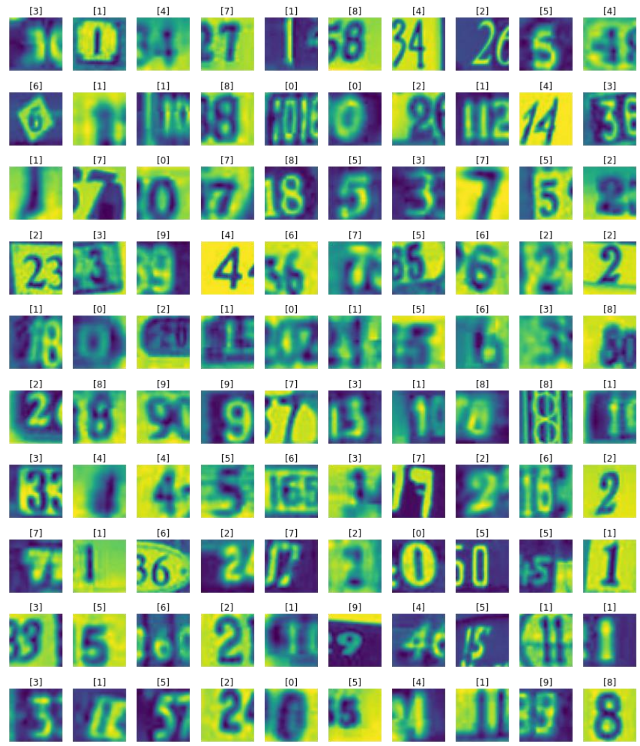
	PCA	Siamese with triplet loss
Training time (s)	307	1297
Inference time (s)	55 (transforming data) + 2 = 57	2.5

TABLE 2 PCA AND SIAMESE: TRAINING AND INFERENCE TIME COMPARISON

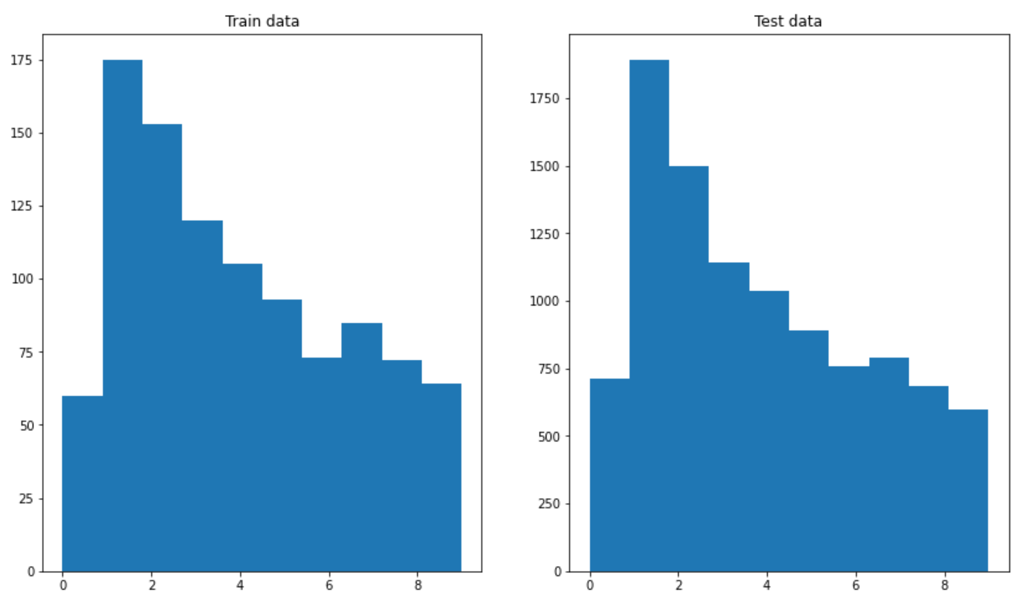
The Siamse model is much more effective, 23 times smaller than PCA's. While the training time of Siamese model is large, it is not an important factor due to the similar reasons discussed in Problem 1: **4 Comparison & Evaluation**: it is a one-time cost, inference time is more focused in real world problems, the correlation with train data, hardware and so on.

Appendixes

1. Sample images in gray-scale



2. Data distribution



3. Hardware specification

Device specifications	
Device name	DuyAnh
Processor	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
Installed RAM	8.00 GB (7.82 GB usable)
Device ID	2ADB9DCA-FCAD-4765-8657-5EE0572E2628
Product ID	00327-35847-67995-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

4. The last 9 layers of original pre-trained model is changed from (a) to (b):

(a)

```
flatten_6 (Flatten)          (None, 1568)              0
dense_16 (Dense)             (None, 1024)              1606656
batch_normalization_27 (Batc (None, 1024)              4096
activation_27 (Activation)   (None, 1024)              0
dropout_10 (Dropout)         (None, 1024)              0
dense_17 (Dense)             (None, 256)               262400
batch_normalization_28 (Batc (None, 256)               1024
activation_28 (Activation)   (None, 256)               0
dropout_11 (Dropout)         (None, 256)               0
dense_18 (Dense)             (None, 10)                2570
```

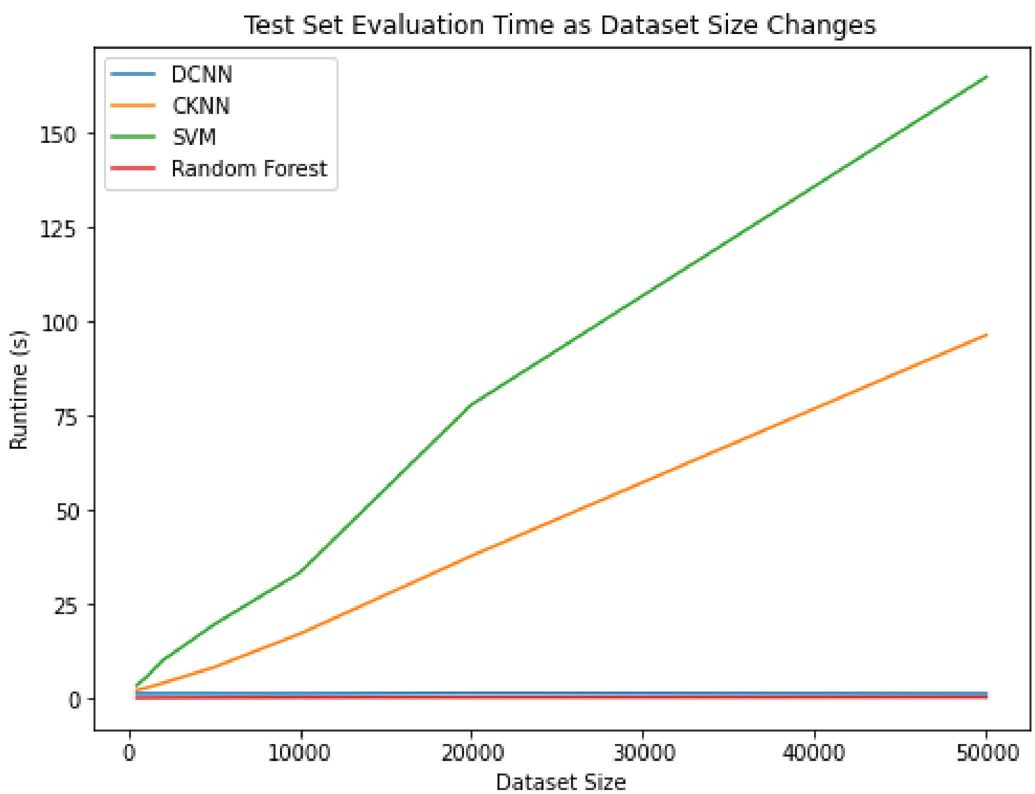
(b)

```
flatten_6 (Flatten)          (None, 1568)              0
dense (Dense)                (None, 256)               401664
dropout (Dropout)            (None, 256)               0
dense_1 (Dense)              (None, 64)                16448
dense_2 (Dense)              (None, 10)                650
```

With the settings are as below:

```python
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(64, activation='relu')(x)
x = layers.Dense(10, activation='softmax')(x)
```

5.



6.

Problem 1: Model 1 Classification report

	precision	recall	f1-score	support
0	0.76	0.71	0.73	711
1	0.79	0.93	0.86	1894
2	0.88	0.84	0.86	1497
3	0.72	0.74	0.73	1141
4	0.80	0.82	0.81	1035
5	0.74	0.72	0.73	892
6	0.74	0.69	0.71	758
7	0.87	0.78	0.82	789
8	0.72	0.61	0.66	683
9	0.69	0.69	0.69	600
accuracy			0.78	10000
macro avg	0.77	0.75	0.76	10000
weighted avg	0.78	0.78	0.78	10000

Problem 1: Model 2 Classification report

	precision	recall	f1-score	support
0	0.84	0.73	0.78	711
1	0.81	0.91	0.86	1894
2	0.89	0.84	0.86	1497
3	0.71	0.77	0.74	1141
4	0.82	0.78	0.80	1035
5	0.67	0.84	0.75	892
6	0.80	0.61	0.69	758
7	0.76	0.85	0.81	789
8	0.67	0.60	0.63	683
9	0.83	0.62	0.71	600
accuracy			0.78	10000
macro avg	0.78	0.75	0.76	10000
weighted avg	0.79	0.78	0.78	10000

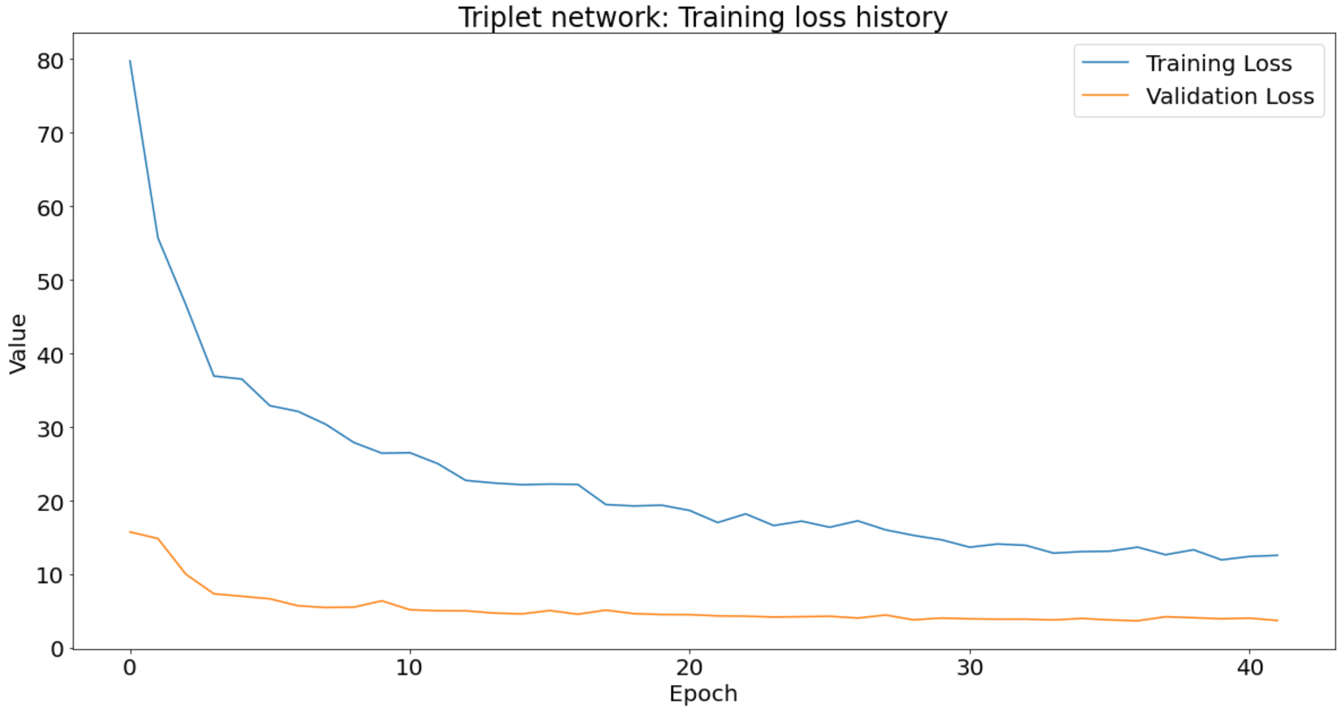
Problem 1: Model 3 Classification report

```
Epoch 00088: val_accuracy did not improve from 0.80000
              precision    recall  f1-score   support

           0       0.72      0.81      0.76       711
           1       0.86      0.89      0.87      1894
           2       0.87      0.90      0.88      1497
           3       0.78      0.75      0.76      1141
           4       0.84      0.77      0.80      1035
           5       0.77      0.81      0.79       892
           6       0.72      0.73      0.73       758
           7       0.82      0.84      0.83       789
           8       0.79      0.67      0.72       683
           9       0.74      0.66      0.70       600

    accuracy                           0.81     10000
   macro avg       0.79      0.78      0.79     10000
weighted avg       0.81      0.81      0.80     10000
```

7. Base network of Siamese network

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 64, 32, 3)]       0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 64, 32, 8)         224
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 64, 32, 8)         584
_________________________________________________________________
batch_normalization_12 (Batc (None, 64, 32, 8)         32
_________________________________________________________________
activation_12 (Activation)   (None, 64, 32, 8)         0
_________________________________________________________________
spatial_dropout2d_9 (Spatial (None, 64, 32, 8)         0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 32, 16, 8)         0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 32, 16, 16)        1168
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 32, 16, 16)        2320
_________________________________________________________________
batch_normalization_13 (Batc (None, 32, 16, 16)        64
_________________________________________________________________
activation_13 (Activation)   (None, 32, 16, 16)        0
_________________________________________________________________
spatial_dropout2d_10 (Spatia (None, 32, 16, 16)        0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 16, 8, 16)         0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 16, 8, 32)         4640
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 16, 8, 32)         9248
_________________________________________________________________
batch_normalization_14 (Batc (None, 16, 8, 32)         128
_________________________________________________________________
activation_14 (Activation)   (None, 16, 8, 32)         0
_________________________________________________________________
spatial_dropout2d_11 (Spatia (None, 16, 8, 32)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               1048832
_________________________________________________________________
batch_normalization_15 (Batc (None, 256)               1024
_________________________________________________________________
activation_15 (Activation)   (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 32)                8224
=================================================================
```

8.


Triplet network: Training loss history
Training Loss
Validation Loss
Value
Epoch
0
10
20
30
40
50
60
70
80

References

Brownlee, J. (2017, July 2). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Machine Learning Mastery. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Mujtaba, H. (2020, September 28). What is Resnet or Residual Network | How Resnet Helps? GreatLearning Blog: Free Resources What Matters to Shape Your Career! https://www.mygreatlearning.com/blog/resnet/

Scikit-learn.org. (2022). Retrieved 10 May 2022, from https://scikit-learn.org.