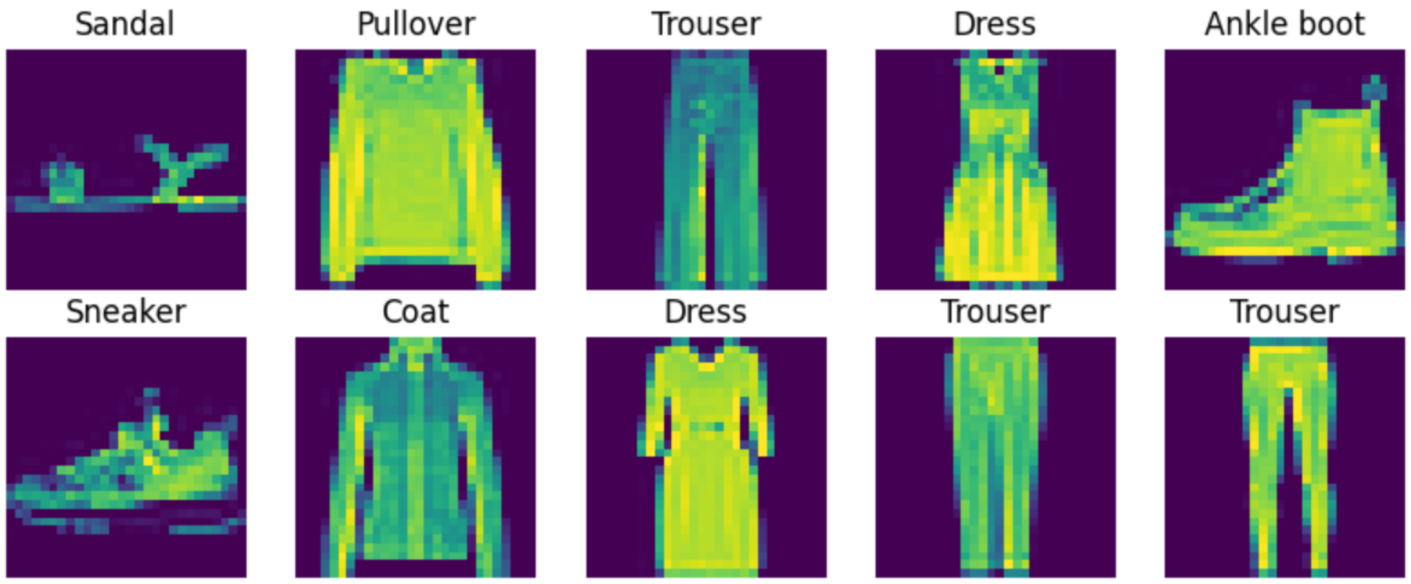
University of Colorado Boulder
CSCI 5922: Neural Nets and Deep Learning
Prof. Danna Gurari

Amulya Ambati
Lab1
February 26, 2024

Q1

Introduction

The MNIST Fashion dataset is a popular benchmark dataset in the field of machine learning and computer vision. It is often used for tasks such as image classification and deep learning model evaluation. This dataset is similar in structure to the original MNIST dataset (handwritten digits), but instead of digits, it consists of grayscale images of various fashion items. It consists of 60,000 training images and 10,000 testing images, each with a resolution of 28x28 pixels. The images are grayscale and depict 10 different classes of fashion items.



The Dataset is loaded from the torchvision module (Link to the dataset).

Train-Test Set

The dataset already comes with a train-test split. Performing another split on the training dataset to obtain some validation data, which can be further used in deciding the best model and hyper-parameter tuning.

Performed a train-validation split of 85-15 % on given train data. The size of the data post-split is as follows:

Train	51,000
validation	9,000
Test	10,000

Data Processing

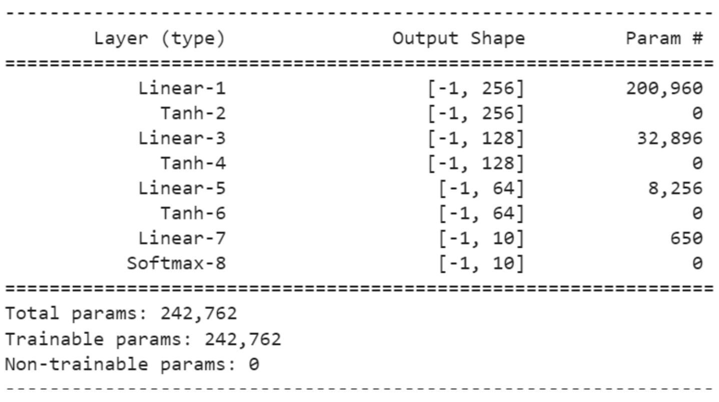
As the dataset is very large, the system might not be able to load the whole dataset at once. Hence training is performed on mini batches which is generally called mini-batch gradient descent.
To facilitate the above process of data loading, a dataloader was created which transforms the data into batches, each of size equal to batch size which is 64 in this case. Since pytorch framework accepts data to be in a tensor format, the pillow images present in the dataset were normalized and then converted into a tensor format.
Since each of our images is of size 28*28 (2d data), and we are designing a fully connected neural network, we are flattening the data into 1d of 784.
The target variable is also one-hot encoded with 10 unique classes, resulting in a 1d tensor of 10.

Modeling

Various experiments were performed by using different optimizers, and different regularization techniques to understand how training or model performance is impacted by the above factors.
A fully connected neural network is considered with a loss function of cross-entropy loss since we are performing classification techniques and maximizing the predicted softmax probabilities to be similar to one hot encoded ground truth.
Baseline architecture consists of three dense layers of size 256, 128, and 64 with an activation function of 'tanh'. The output layer is of size 10 with an activation function of softmax.
Few of the constant model parameters were

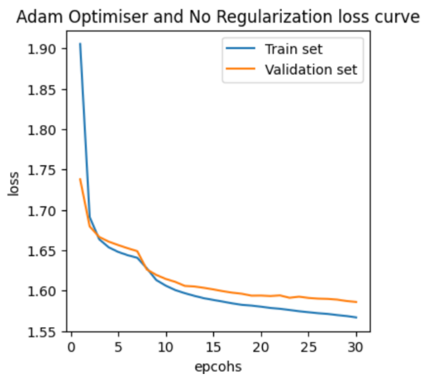
- loss: cross-entropy loss
- batch size: 64
- epochs: 30

1) Model 1 - No regularization & Adam Optimizer

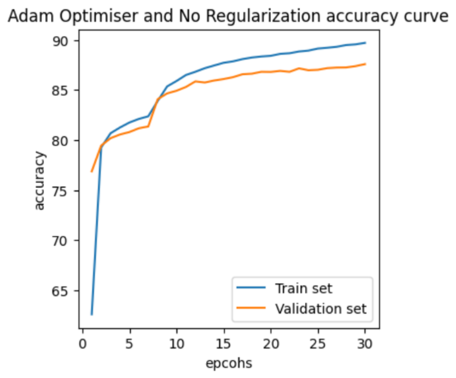


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]         200,960
              Tanh-2                  [-1, 256]               0
            Linear-3                  [-1, 128]          32,896
              Tanh-4                  [-1, 128]               0
            Linear-5                   [-1, 64]           8,256
              Tanh-6                   [-1, 64]               0
            Linear-7                   [-1, 10]             650
           Softmax-8                   [-1, 10]               0
================================================================
Total params: 242,762
Trainable params: 242,762
Non-trainable params: 0
----------------------------------------------------------------
```

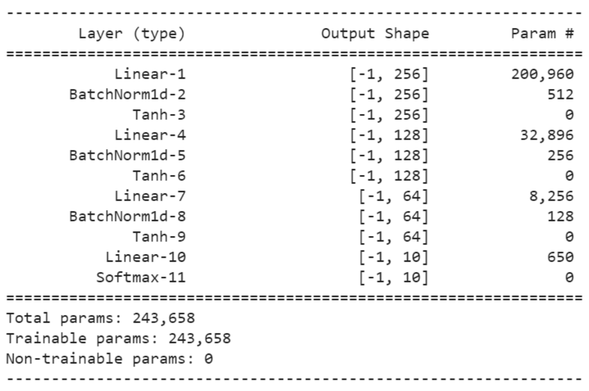
The loss/learning curve is as follows:



The accuracy curve is as follows:

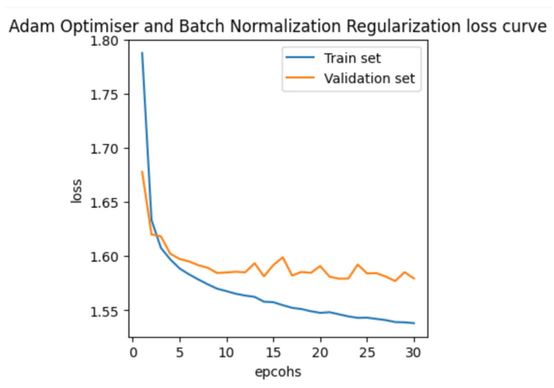


2) Model 2 - Batch Normalization Regularization & Adam Optimizer

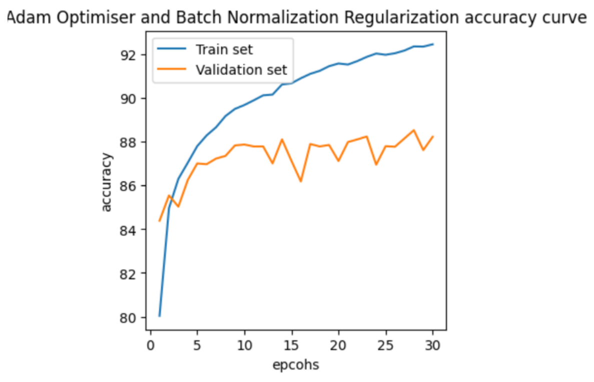


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]         200,960
       BatchNorm1d-2                  [-1, 256]             512
              Tanh-3                  [-1, 256]               0
            Linear-4                  [-1, 128]          32,896
       BatchNorm1d-5                  [-1, 128]             256
              Tanh-6                  [-1, 128]               0
            Linear-7                   [-1, 64]           8,256
       BatchNorm1d-8                   [-1, 64]             128
              Tanh-9                   [-1, 64]               0
           Linear-10                   [-1, 10]             650
          Softmax-11                   [-1, 10]               0
================================================================
Total params: 243,658
Trainable params: 243,658
Non-trainable params: 0
----------------------------------------------------------------
```

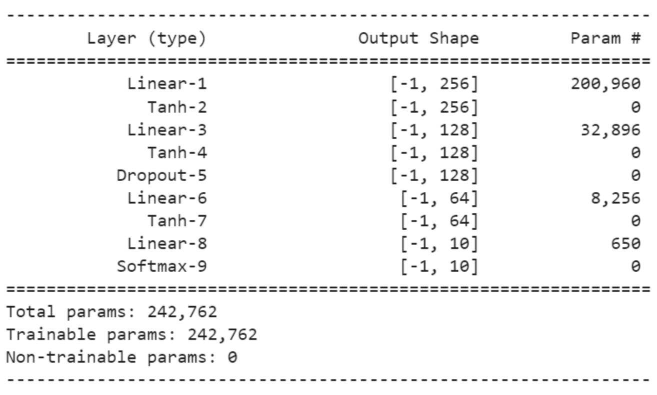
The loss/learning curve is as follows:



The accuracy curve is as follows:

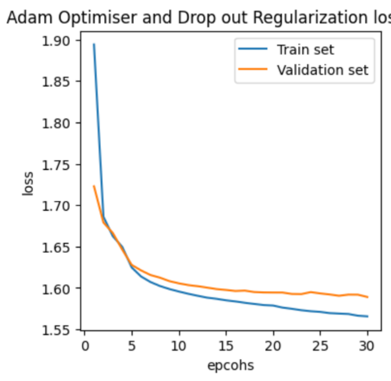


3) Model 3 - Drop out regularization & Adam Optimizer

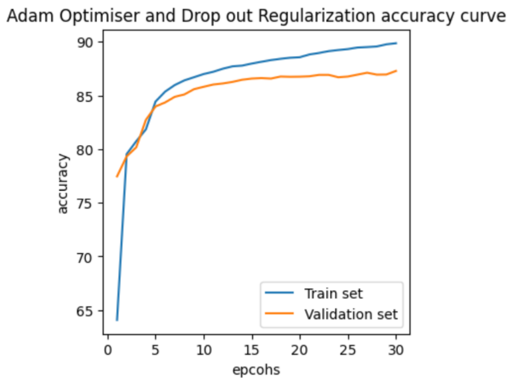


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]         200,960
              Tanh-2                  [-1, 256]               0
            Linear-3                  [-1, 128]          32,896
              Tanh-4                  [-1, 128]               0
           Dropout-5                  [-1, 128]               0
            Linear-6                   [-1, 64]           8,256
              Tanh-7                   [-1, 64]               0
            Linear-8                   [-1, 10]             650
           Softmax-9                   [-1, 10]               0
================================================================
Total params: 242,762
Trainable params: 242,762
Non-trainable params: 0
----------------------------------------------------------------
```

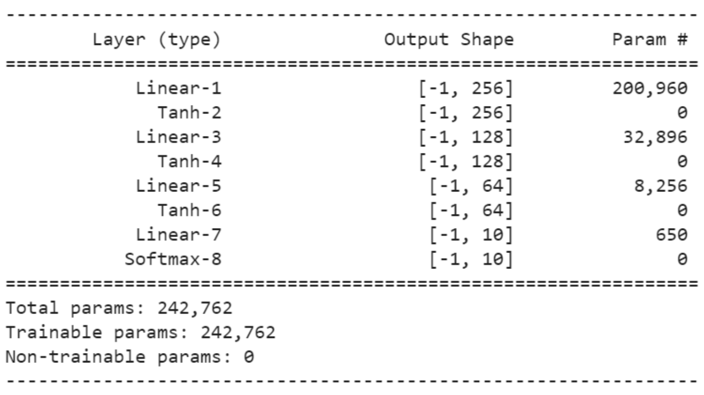
The loss/learning curve is as follows:



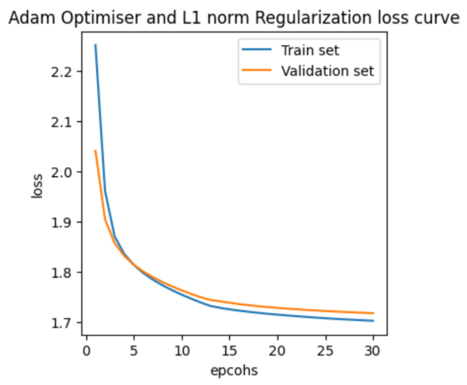
The accuracy curve is as follows:



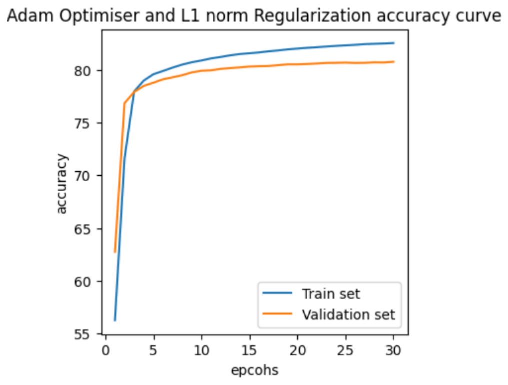
4) Model 4 - L1-Norm regularization & Adam Optimizer


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]         200,960
              Tanh-2                  [-1, 256]               0
            Linear-3                  [-1, 128]          32,896
              Tanh-4                  [-1, 128]               0
            Linear-5                   [-1, 64]           8,256
              Tanh-6                   [-1, 64]               0
            Linear-7                   [-1, 10]             650
           Softmax-8                   [-1, 10]               0
================================================================
Total params: 242,762
Trainable params: 242,762
Non-trainable params: 0
----------------------------------------------------------------
```

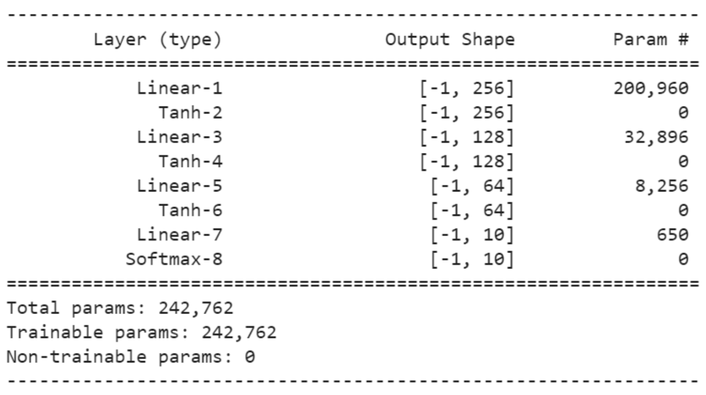
The loss/learning curve is as follows:



The accuracy curve is as follows:

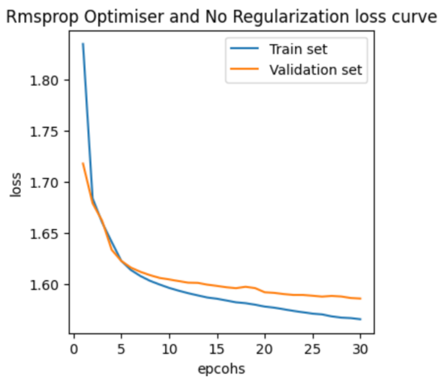


5) Model 5 - No regularization & Rmsprop Optimizer

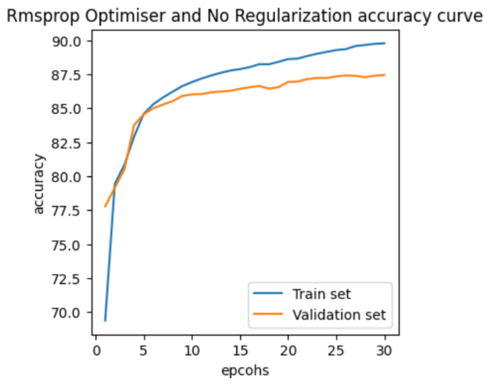


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]         200,960
              Tanh-2                  [-1, 256]               0
            Linear-3                  [-1, 128]          32,896
              Tanh-4                  [-1, 128]               0
            Linear-5                   [-1, 64]           8,256
              Tanh-6                   [-1, 64]               0
            Linear-7                   [-1, 10]             650
           Softmax-8                   [-1, 10]               0
================================================================
Total params: 242,762
Trainable params: 242,762
Non-trainable params: 0
----------------------------------------------------------------
```

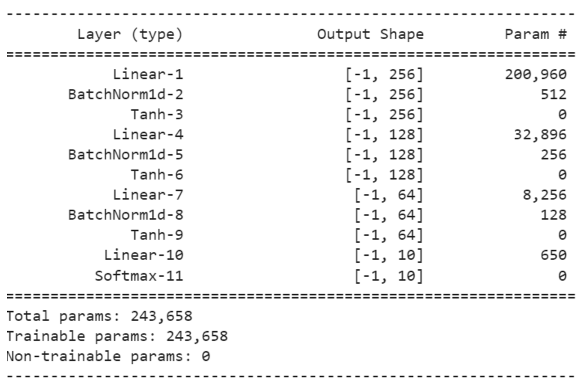
The loss/learning curve is as follows:



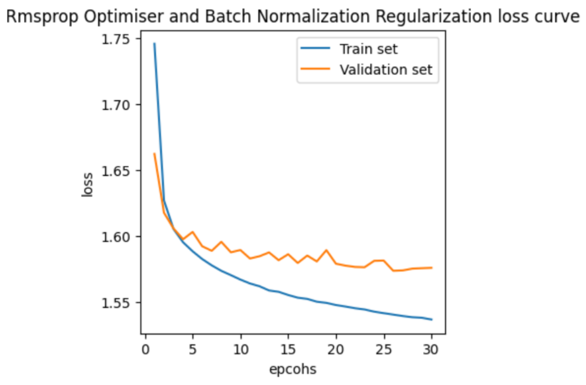
The accuracy curve is as follows:



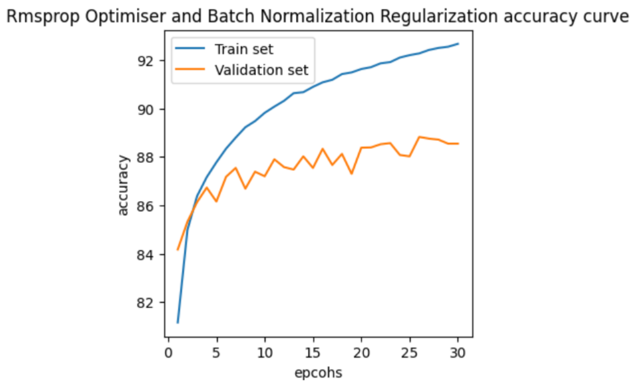
6) Model 6 - Batch Normalization Regularization & Rmsprop Optimizer


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]         200,960
       BatchNorm1d-2                  [-1, 256]             512
              Tanh-3                  [-1, 256]               0
            Linear-4                  [-1, 128]          32,896
       BatchNorm1d-5                  [-1, 128]             256
              Tanh-6                  [-1, 128]               0
            Linear-7                   [-1, 64]           8,256
       BatchNorm1d-8                   [-1, 64]             128
              Tanh-9                   [-1, 64]               0
           Linear-10                   [-1, 10]             650
          Softmax-11                   [-1, 10]               0
================================================================
Total params: 243,658
Trainable params: 243,658
Non-trainable params: 0
----------------------------------------------------------------
```

The loss/learning curve is as follows:



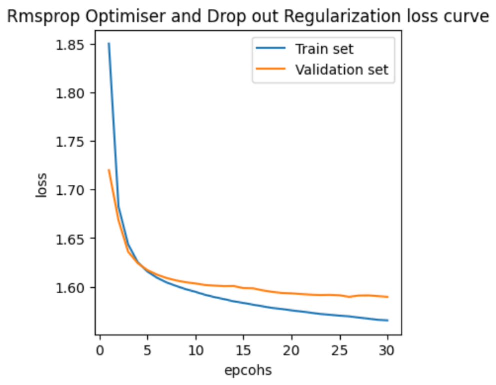
The accuracy curve is as follows:



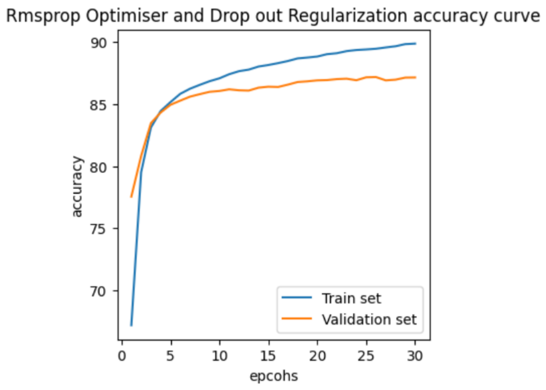
7) Model 7 - Drop out regularization & Rmsprop Optimizer

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]         200,960
              Tanh-2                  [-1, 256]               0
            Linear-3                  [-1, 128]          32,896
              Tanh-4                  [-1, 128]               0
           Dropout-5                  [-1, 128]               0
            Linear-6                   [-1, 64]           8,256
              Tanh-7                   [-1, 64]               0
            Linear-8                   [-1, 10]             650
           Softmax-9                   [-1, 10]               0
================================================================
Total params: 242,762
Trainable params: 242,762
Non-trainable params: 0
----------------------------------------------------------------
```

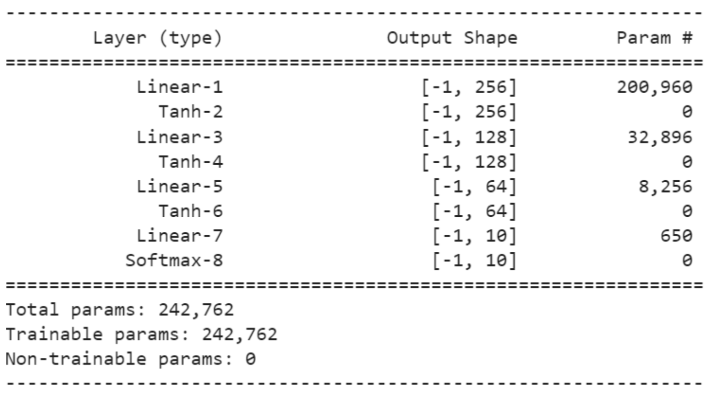
The loss/learning curve is as follows:



The accuracy curve is as follows:

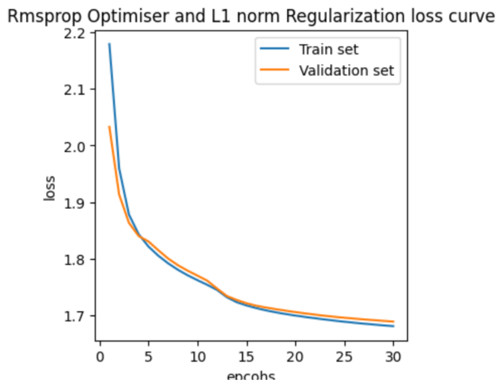


8) Model 8 - L1-Norm Normalization Regularization & Rmsprop Optimizer

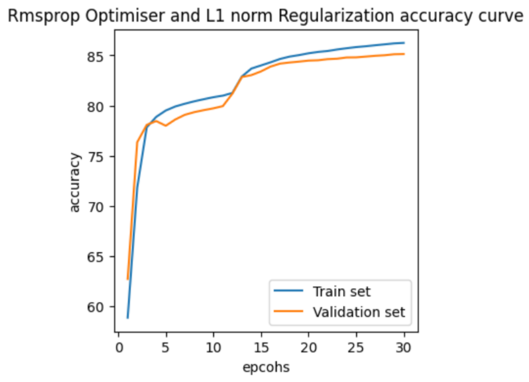


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]         200,960
              Tanh-2                  [-1, 256]               0
            Linear-3                  [-1, 128]          32,896
              Tanh-4                  [-1, 128]               0
            Linear-5                   [-1, 64]           8,256
              Tanh-6                   [-1, 64]               0
            Linear-7                   [-1, 10]             650
           Softmax-8                   [-1, 10]               0
================================================================
Total params: 242,762
Trainable params: 242,762
Non-trainable params: 0
----------------------------------------------------------------
```

The loss/learning curve is as follows:



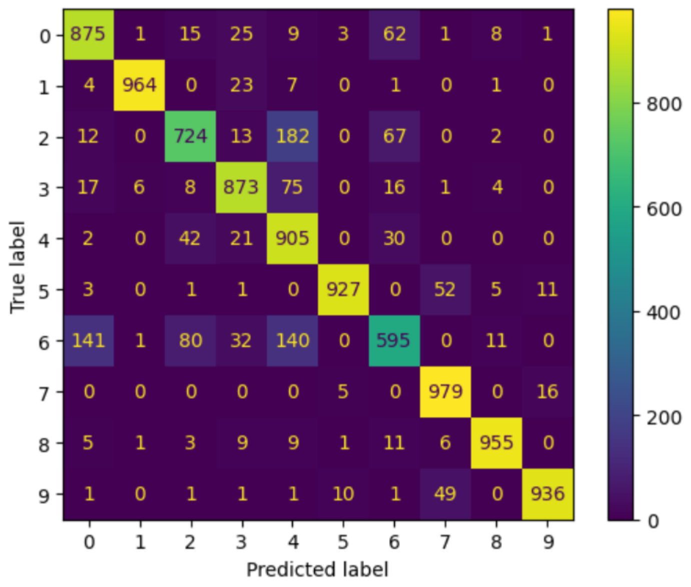
The accuracy curve is as follows:



Post comparing all these models, a model with batch normalization technique has better performance. Hence finalizing the model with rmsprop optimizer and batch normalization regularization technique and retraining the model on a bigger set of train + validation sets.
The performance of the final model is as follows:

Data	Accuracy	loss
Train	95.08	1.51
Test	87.33	1.58

Confusion matrix on test set, allows us to see which classes are underperforming and are frequently getting mispredicted as other classes.



We can see that class 6 which is 'Shirt' is often getting mispredicted as Class 0 which corresponds to 'T-shirt/top' and class 4 which corresponds to 'Coat'. This is likely because shirts, T-shirts, and coats look very similar, and differentiating them needs much richer features/ architecture.

Observations/Patterns

Optimizer	Regularization technique	Time taken(in seconds)	Train accuracy	Validation accuracy	Train l
Adam	"	67.89	89.69	87.57	1.56
Adam	Batch normalization	75.66	92.46	88.23	1.53
Adam	Drop out	64.70	89.69	87.27	1.56
Adam	L1- Norm	89.51	89.69	80.76	1.7
Rmsprop	"	55.44	89.78	87.45	1.56
Rmsprop	Batch normalization	64.91	92.66	88.54	1.57
Rmsprop	Drop out	56.36	89.87	87.14	1.56
Rmsprop	L1- Norm	78.38	89.24	85.13	1.68

If we compare the time taken for training the models, in general, we can see that the Adam optimizer takes a little longer when compared to the Rmsprop optimizer, and regularization techniques like batch normalization and l1- norm take more time than without any regularization and drop out, because extra computation is happening there. However, the difference is minimal and can be overlooked for better performance in terms of accuracy.

While comparing accuracies, a model with batch normalization has the highest accuracy for both Adam and rmsprop optimizers, but we can see higher fluctuations in the loss curve and accuracy curve when compared to other models.

The loss for l1-norm is in general higher when compared to other models since we explicitly add penality terms of modulus of learnable parameters.

Q2

Introduction

The data for this analysis was sourced from Kaggle (Link to the dataset). This dataset contains a total of 989 train images and 140 test images with 10 unique categories of fractures which are avulsion fractures, comminuted fractures, fracture-dislocations, greenstick fractures, hairline fractures, impacted fractures, longitudinal fractures, oblique fractures, pathological fractures, spiral fractures.

Train-Test Set

The dataset itself came with a train-test split. Hence no further split was performed in this case.

Train	989
Test	140

Data Processing

Even though the dataset is small mini-batch gradient descent was performed since it allows for more gradient updates.

To facilitate the above process of data loading, a dataloader was created which transforms the data into batches, each of size equal to batch size which is 32 in this case. Since pytorch framework accepts data to be in a tensor format, the images present in the dataset were converted into a tensor format.

We will be using a CNN architecture, the data format is retained in a 2d structure.

The target variable is also one-hot encoded with 10 unique classes, resulting in a 1d tensor of 10.

Modeling

The aim here was to load a pre-trained model and fine-tune it according to our dataset since pre-trained models are trained on huge datasets the initial layers contain information which are not data specific and instead contain kernels on how to extract rich features irrespective of data.

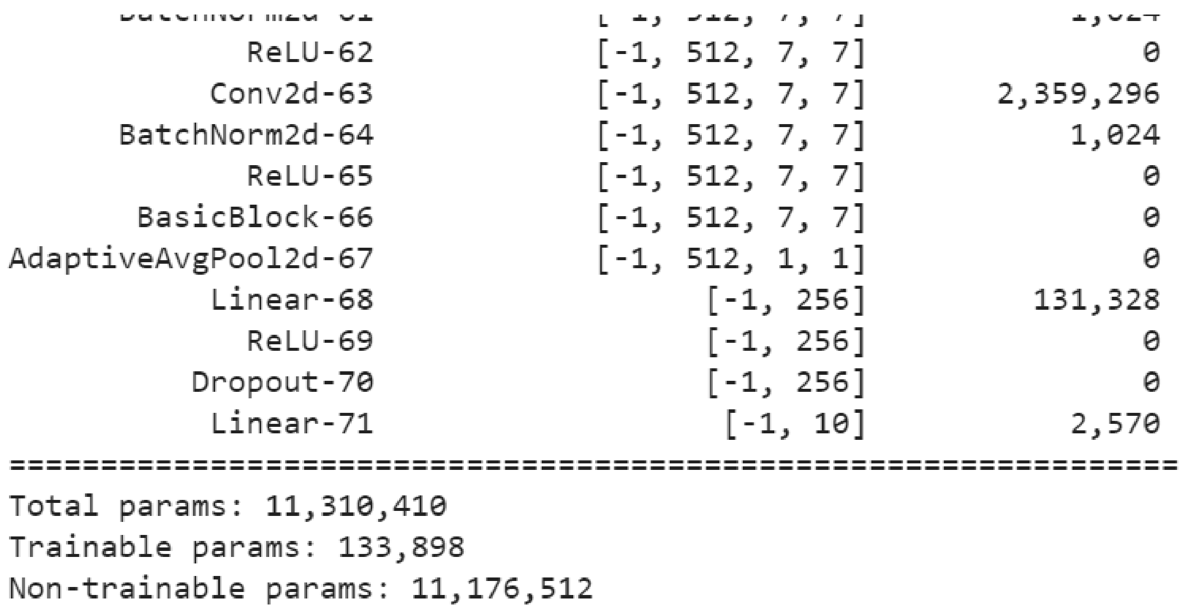
For this experiment, ResNet18, one of the start-of-the art models was considered which was pre-trained on the image-net dataset spanning 1000 classes for transfer learning. Except for the final output layer all other layers are retained and an extra dense layer of size 256 is added, before adding our custom output layer of size 10. Moreover, for the layers coming from the pre-trained model, the parameters are configured as nontrainable ensuring that the learned weights remain static throughout the training phase and only newly added dense layers parameters were updated.

· loss: cross-entropy loss

· batch size: 32

· epochs: 10

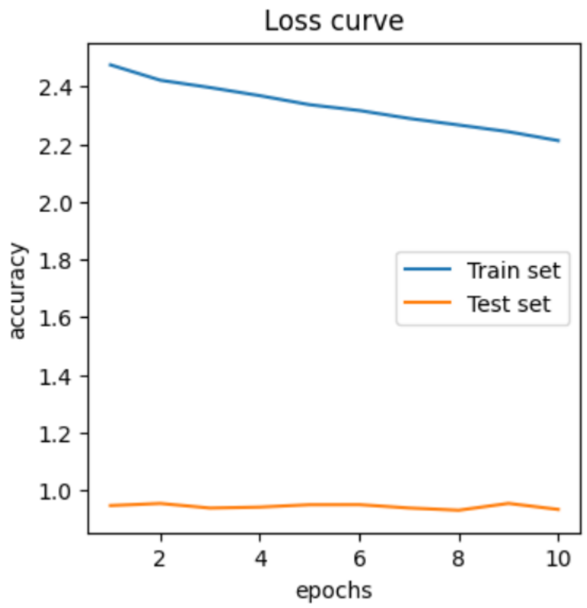
· Optimizer: SGD with lr=0.001, momentum=0.9

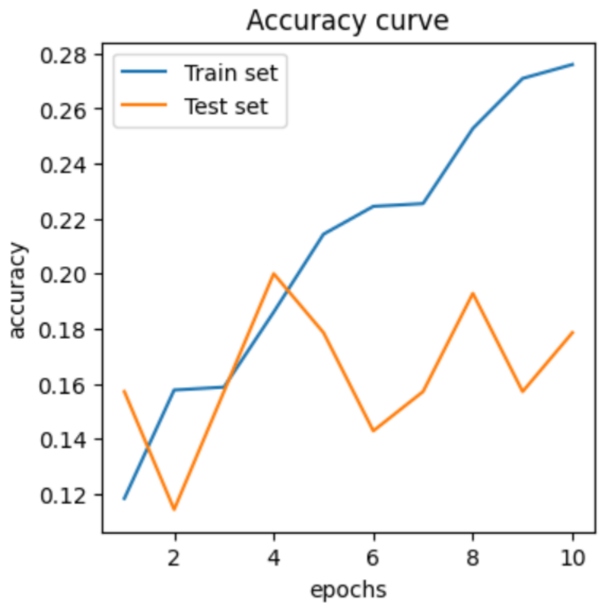


```
           ReLU-62            [-1, 512, 7, 7]               0
         Conv2d-63            [-1, 512, 7, 7]       2,359,296
    BatchNorm2d-64            [-1, 512, 7, 7]           1,024
           ReLU-65            [-1, 512, 7, 7]               0
     BasicBlock-66            [-1, 512, 7, 7]               0
AdaptiveAvgPool2d-67          [-1, 512, 1, 1]               0
         Linear-68                  [-1, 256]         131,328
           ReLU-69                  [-1, 256]               0
        Dropout-70                  [-1, 256]               0
         Linear-71                   [-1, 10]           2,570
================================================================
Total params: 11,310,410
Trainable params: 133,898
Non-trainable params: 11,176,512
```

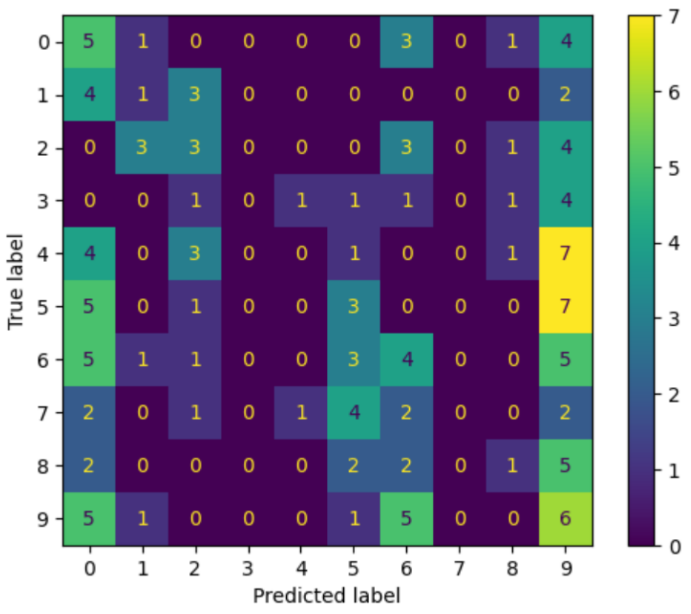
All the layers come from ResNet architecture, and layers 68 to 71 are added and finetuned as per our dataset.

The training went as follows:





Confusion matrix on test set, allows us to see which classes are underperforming and are frequently getting mispredicted as other classes.



We can see that, the predictions are very random and don't hold water.

The loss is reducing gradually, hence had we run the model for a longer duration, we can probably expect further reduction in loss and improvement in accuracy.

Data	Accuracy	loss
Train	27.60	2.21
Test	17.86	0.933

Observations/Patterns

According to my understanding, for this dataset the pre-trained models weren't a huge plus, because we are seeing lower accuracies of around 10-20 %, this might be because the current dataset is very different from the dataset the pre-trained model was trained on. However, the pre-trained model might have helped in extracting relevant features in the initial epochs itself, and had there not been a pre-trained model, initial epoch accuracies might have been much lower.