
Multiclass Classification for Fashion MNIST using Deep Learning

Abhilash Kasarla, Harish Prasanth, Inderjot Singh, Saideep Reddy, Vaishali Rajendren
ECE 285 : Deep Learning for Image Processing

Abstract

Multiclass image classification is advancing rapidly albeit with its limitations and challenges. One major limitation is the need for large annotated dataset for building robust classifiers. In this project we implement a classification pipeline for Fashion-MNIST dataset, a better alternative to MNIST dataset. We incrementally test architectures of different complexities to establish baseline and improve upon it. Further we implement techniques like early-stopping, batch-normalization, data-augmentation and boosting. Finally, we use transfer learning to fine tune a model trained on a much larger and generic dataset (ImageNet for our case) and show that we can achieve reasonable accuracy even when the size of dataset is reduced to one-tenth of the original

1 Introduction

Image classification is an important problem that forms the basis for many computer vision tasks. Before deep learning, traditional computer vision methods, that employed pre-defined filters for extracting useful information from images, were used for this task. On the other hand, Deep Learning learns these filters and builds hierarchy of them to learn rich descriptors for the data. This has been significantly successful especially in vision related tasks and as a result almost all state-of-art results in vision are achieved using deep learning based architectures. This growth has been driven by (i) increase in computational power, (ii) better algorithms, and most importantly (iii) large amount of data. Data is now looked upon as a resource and is only going to grow as data driven technology (home automation devices, IoT etc) becomes more common.

1.1 Motivation

A major challenge that deep learning faces is the huge amount of annotated data required to train the network for supervised applications. Most networks trained on a small dataset tend to generalize very badly i.e they perform poorly on data that the network has not seen before. One of the proposed solution is called Transfer Learning ref[Distilling the Knowledge...Hinton] that is based on the idea that the initial layers of the network tend to learn features that are common across images from different datasets. For our project we use a tractable dataset called Fashion-MNIST which has 60000 training samples and 10000 testing samples of $28 * 28$ gray scale images, where each image is associated with 1 out of 10 classes.

The classes are:

0 T-shirt/top	1 Trouser	2 Pullover	3 Dress	4 Coat
5 Sandal	6 Shirt	7 Sneaker	8 Bag	9 Ankle boot

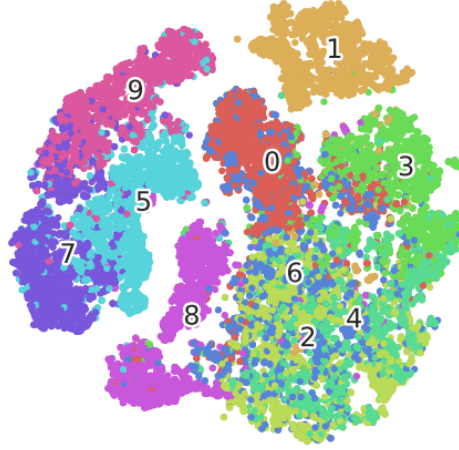


Figure 2.2.2 t-SNE Visualization of Fashion-MNIST

MNIST is considered a solved problem now with modern architectures matching human accuracy on classifying digits. Fashion-MNIST on the other hand still has lot of scope of improvement as it is more complex than MNIST but at the same time is similar to MNIST in the sense that it is tractable and doesn't use up lot of memory/computational resources.

2 Description of Method

2.1 Algorithm

To establish baseline performance we train a LeNet-5 architecture from scratch on the data. We then use Early-stopping, Batch-Normalization and boosting to improve the performance of our model. We also plot the validation and training loss per epoch as well as the validation accuracy per epoch. We then use two extremely popular architectures namely VGG-16 and Resnet-18 pre-trained on the Imagenet dataset, and fine tune the network for our data.

2.2 Architecture

2.2.1 LeNet-5:

The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier. The input of LeNet-5 passes through the first convolutional layer with 6 feature maps or filters having size 55 and a stride of one and then through the average pooling layer with a filter size 22 and a stride of two. Next, there is a second convolutional layer with 16 feature maps having the size of 55 and a stride of 1 and followed by a similar average pooling layer as the first except it has 16 feature maps. The third layer is a fully connected convolutional layer with 120 feature maps and the fourth layer is a fully connected layer with 84 units. Finally, there is a fully connected softmax output layer with 10 possible values from 0 to 9.

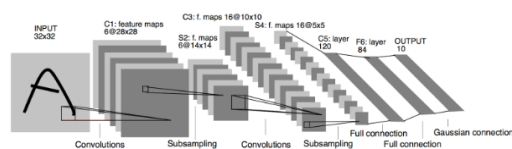


Figure 2.2.2 LeNet-5

2.2.2 Boosted LeNet-5:

The next thing that we tried was to apply the traditional boosting technique to deep learning. We trained three LeNet-5's one after another. The first network was trained as usual on the whole training set whereas the second network was trained on a mixture of patterns containing 50% of which the first net got it right and 50% of which it got wrong. The third network was trained on the patterns on which the first and the second networks disagree. During testing the outputs of 3 networks are simply added.

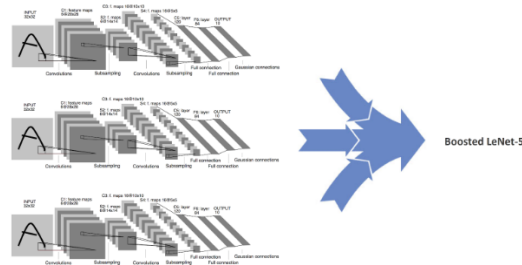


Figure 2.2.2 Boosted LeNet-5

2.2.3 Boosted Network with 2 Conv Layers + 3 FC layers:

We built a variant of LeNet with more channels along with batch normalization, dropout and boosting. The architecture had 64 channels in the first convolutional layer, 512 in the second and both layers had batch normalization along with max pooling. Three fully connected layers followed the convolutional layers with dropout. The results were better than the previous results with 92% accuracy on the test data. As done in the previous network we tried boosting (training on 3 such networks) but the accuracy marginally improved (92.2%) on the test data.

2.2.4 VGG-16 (Batch Normalized):

VGG-16 is a deep network with 13 convolutional layers followed by 3 FC layers. Each convolutional layer uses only 3*3 kernels and is followed by max pooling, batch norm and then ReLU. The fully connected layers are implemented along with dropout ($p=0.5$). We made use of Transfer Learning technique, it is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems. We took the pretrained VGG-16 model which was trained on Imagenet, and dropped the last Fully connected layer, inserted our own layer and then trained only the Fully connected network keeping the Convolutional network freezed. In this way we are making use of the knowledge the network has gained or the basic low level features that the network has learned from a very large dataset such as Imagenet.

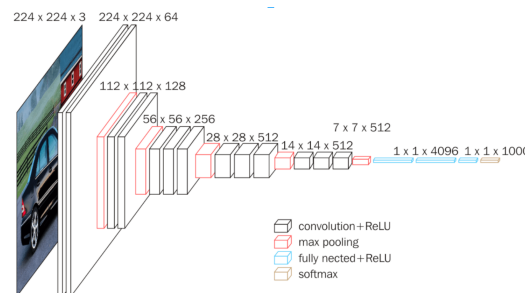


Figure 2.2.4 VGG-16 Network

2.2.5 ResNet-18:

ResNet-18 is the network that introduced the idea of residual learning. The network is based on the concept that a deep network with residual connections has to perform better than a smaller network

as the rest of the network can emulate identity function to give the exact same result as that of the less deeper network. The network also helps eliminate the problem of vanishing gradients and allow the gradients to pass through the residual pathway. The variant of ResNet we used is called ResNet-18 and as the name suggests has 18 layers (other variants include ResNet-50, ResNet-101 and ResNet-152)

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	$7 \times 7, 64, \text{stride } 2$
conv2_x	$56 \times 56 \times 64$	$3 \times 3 \text{ max pool, stride } 2$ $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	$7 \times 7 \text{ average pool}$
fully connected	1000	$512 \times 1000 \text{ fully connections}$
softmax	1000	

Figure 7.2 ResNet-18 Network

The network also implements batch normalization after each convolution layer. The last layer is a FC layer with 1000 units (for ImageNet dataset). As explained previously, we replaced this layer with a 10 unit FC layer for classifying our dataset. We also used horizontal flipping for data augmentation. The goal of this experiment (transfer learning) is to use a deep and computationally intensive to train network pre-trained on a larger dataset (ImageNet in this case). The generic-descriptors (features learned by first few layers) along with some fine tuning over a smaller dataset gives much better results.

2.3 Equations

Cross Entropy Loss:

$$E = - \sum_{i=1}^M y_{o,c} \log(p_{o,c}) \quad (1)$$

where, M - number of classes; log - the natural log; y - binary indicator (0 or 1) if class label c is the correct classification for observation o; p - predicted probability observation o is of class c

Batch Normalization:

$$\hat{x} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (2)$$

$$\hat{y} \leftarrow \gamma \hat{x} + \beta \equiv BN_{\gamma, \beta}(x_i) \quad (3)$$

where, \hat{x} : input; σ_B^2 : mini-batch variance; μ_B : mini-batch mean; γ and β : parameters to be learned

3 Implementation Details:

Resnet-18/VGG-16 Transfer Learning

- The network expects the image to be of size 224 x 224 so we start by resizing the image
- Two data augmentation methods are used namely Random Crop (which crops the images at random from default 0.08 to 1.0 times the original size) and Random Horizontal flip

- Since the network is pre-trained on RGB images we extend our grayscale images to three channels by stacking single channel thrice. We take 10% of the training data for validation
- Pretrained ResNet-18/VGG-16 model is loaded from torchvision.models and the last layer is replaced by a FC layer with 10 output units for our classification problem.
- The model is then trained using Stochastic Gradient Descent with 0.001 learning rate and 0.9 momentum. Cross-entropy is chosen as the loss criterion.
- We select the % of dataset we want to train on (only for ResNet) and the number of training epochs, which is also dependent on the training set size because of computational and time constraints.
- Validation accuracy and loss is stored after each epoch and Training loss is stored after each batch
- After each epoch, based on the validation accuracy, the best model is saved.
- We compute the test case accuracy of the best model.
- Finally we plot the validation accuracy/loss per epoch and average training loss per batch

4 Experimental setting:

4.1 Dataset:

Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. It is a new dataset comprising of 70,000 (60,000 training and 10,000 testing samples) 28x28 grayscale images of fashion products from 10 categories. It was developed with the intention of replacing the overused MNIST dataset to benchmark modern CV tasks and machine learning algorithms. It is freely available at <https://github.com/zalandoresearch/fashion-mnist>

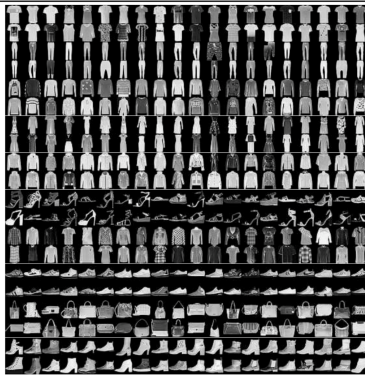
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Figure 7.2 Fashion-MNIST Dataset

4.2 Training parameters:

Boosted Network-2 convolutional layers and 3 FC layers	VGG-16/ResNet-18
Learning rate: 0.001	Learning rate: 0.001, Momentum: 0.9
Number of epochs: 20	Number of epochs: 10-25
Metric: Test Accuracy	Metric: Test Accuracy
Loss: Categorical Cross Entropy	Loss: Categorical Cross Entropy
Adam Optimizer	Loss: Categorical Cross Entropy
Batch size: 64	Batch size: 40
Dropout fraction: 0.35	NA

4.3 Hardware used:

The experiments were performed on a machine with GeForce GTX 1080 Ti 11178 MB GRAM GPU and 16384 MB of regular RAM.

5 Results:

5.1 Figures:

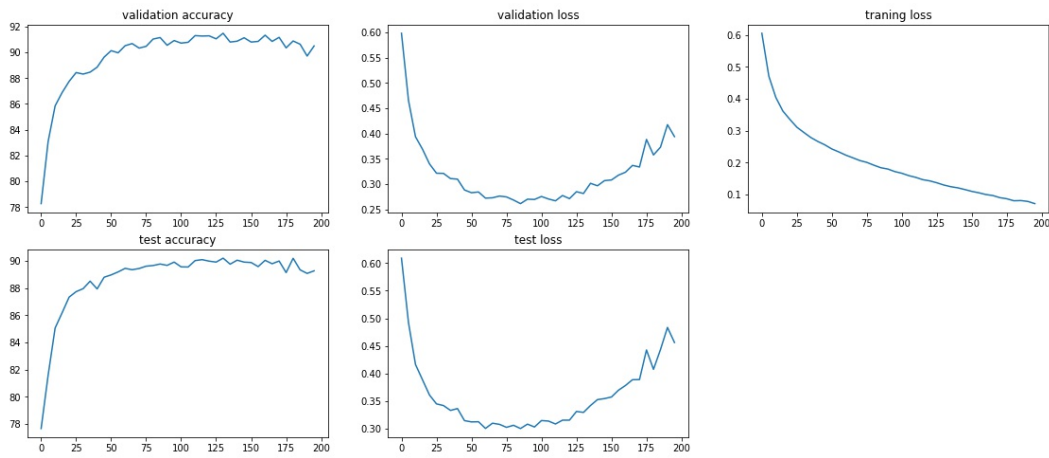


Figure 5.1 LeNet-5 basic network training metrics vs epochs

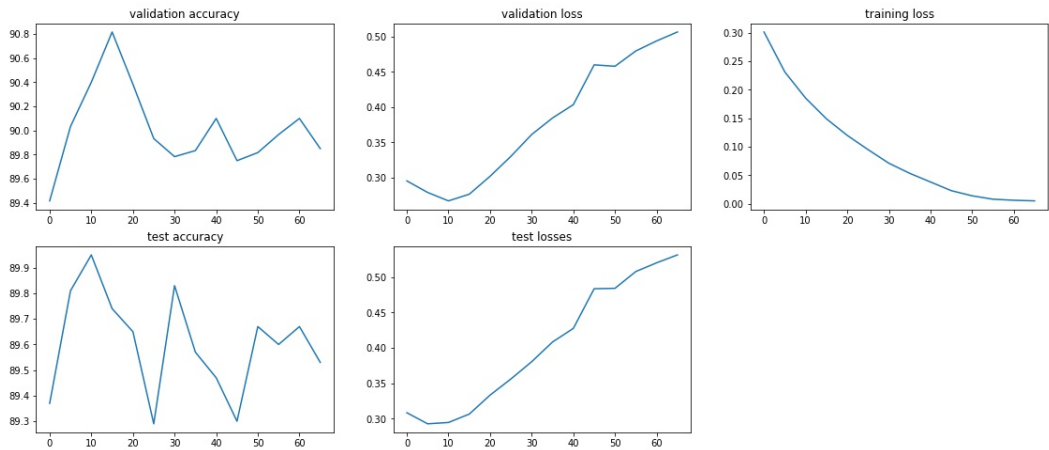


Figure 5.1 LeNet-5 with batch normalization

We can observe that after batch normalization the training converges faster i.e. around 10 epochs compared to 60 epochs without batch normalization.

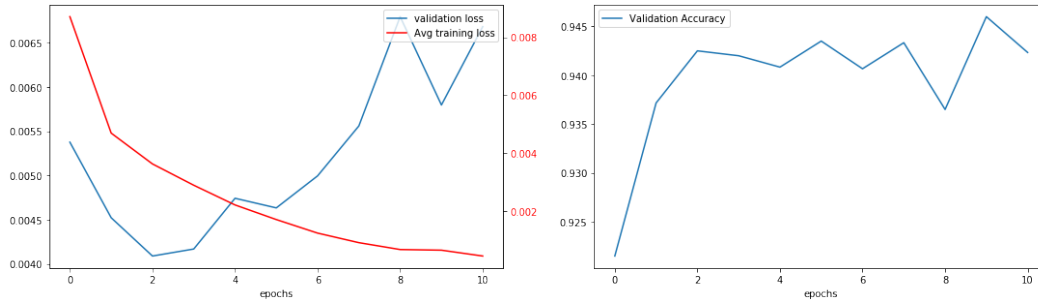


Figure 5.1 VGG-16 without data augmentation

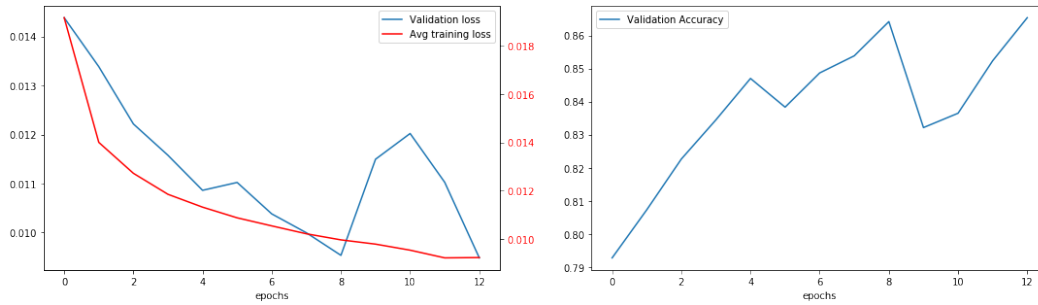


Figure 5.1 VGG-16 With data augmentation - Random Resized Crop and Random Horizontal Flip
Comparatively the one without data augmentation starts overfitting as early as 2nd epoch.

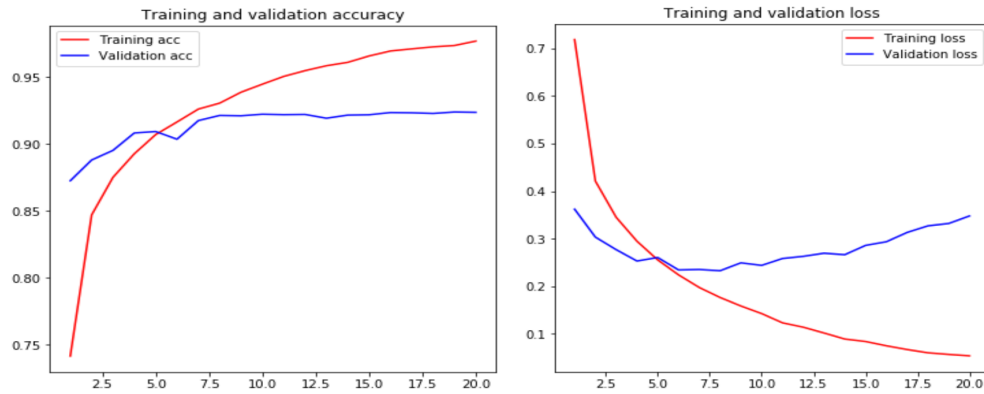


Figure 5.1 2Conv+3FC+Batch Normalization+Dropout

ResNet18: The dataset we have has 54000 training images 6000 validation images and 10000 test images. We initially take just half of the dataset (27000 images).



Figure 5.1 ResNet-18: 10% Training Data (90.69% Test Accuracy)



Figure 5.1 ResNet-18: 50% Training Data (92.57% Test Accuracy)



Figure 5.1 ResNet-18: 100% Training Data (93.44% Test Accuracy)

Through this example, we observe that even while just training on 5400 images we can get a reasonable accuracy of 90.69% on the test set which has 10000 images. This clearly highlights the tremendous benefit one can achieve by distilling the knowledge of a large network and using it for classifying images from a smaller dataset.

5.2 Tables:

Table 1: Architecture and Accuracy

Architecture	Test Accuracy
LeNet-5 (Baseline)	89.87%
Boosted LeNet-5 (3 nets)	90.82%
2Conv+3FC+BatchNormalization+Dropout+Boosting	92.12%

Table 2: VGG16 without data augmentation

Batch size	Epochs	Test Accuracy	Time Taken
30	1	91.95%	927s
10	3	93.11%	3343s
40	7	93.85%	7200s
40	11	93.76%	10750s

Table 3: ResNet18

% of Training data used	Number of Epochs Trained	Best Case Test Accuracy
10	25	90.69%
50	15	92.57%
100	10	94.33%

Table 4: Comparison

Architecture	Number of Epochs Trained	Optimizer	Best Case Test Accuracy
Boosted Network	20	Adam	90.82%
VGG16	7	SGD	93.85 %
ResNet18	10	SGD	94.33%

5.3 Success Cases:

As this is used as a benchmark for the recent machine learning algorithms, there are many architectures that have been tried. There is a list of 129 classifiers built on the dataset (not using deep learning) and the best model among them (by accuracy) is a support vector machine with a polynomial kernel with an accuracy of 89.7%. There is a growing list of benchmarks using deep learning which are yet to be verified. This link has all the architectures and results that are reported.

6 Discussion

6.1 Inference

Based on our experiments with various architectures we can infer that:

- One could see from the plots above batch normalization definitely made the convergence faster requiring less number of iterations/epochs.
- Using early-stopping validation set helps in selecting a model that doesn't overfit on the training data and is robust
- Boosting is an effective method especially for training weak classifiers and combining their knowledge to make predictions
- Results obtained from Transfer learning approach validate the fact that most deep learning architectures today are not trained from scratch
- This ability of deep learning networks to learn information that can be generalized across different cases paves the way for building networks that can perform multiple tasks (multi-task learning) and also provides insights for general intelligence

6.2 Conclusion

We conducted the experiments on various architectures employing different improvements like Dropout, Batch Normalization. We learnt that using a complex network with more parameters which takes care of over-fitting will definitely improve the performance. There are some drawbacks with traditional architectures that are taken care by the newer architectures. There is a scope for lot of improvement with more compute power and storage.

6.3 Learning & Challenges

After analyzing different architectures and results we identify that various improvements could be done:

- To train deeper networks with more parameters, we require higher computing power. So, training on high performance CPU/GPU's would certainly give us better results.
- Run the network with different hyper-parameters. Not just the network, but when we want to tune the hyper parameters, we would require high compute power as these operations are also very compute-intensive

6.4 Future Work

Apart from employing greater computing resource there are few techniques that one could try. Recent paper on DropPruning for Model Compression suggests that without the use of structurally complex architectures one could achieve state-of-the-art results on existing architectures like LeNet-5, VGG-16. Although using networks like ResNet eliminates problems like gradient vanishing, there are other hyper parameters to influence the gradient like the momentum. One could definitely experiment their effects on the performance of the above architectures in the future.

The data augmentation and iterations seemed insufficient in few cases. So, there is a need to experiment and come up with better augmentation techniques to comprehend the results we obtained.

7 Appendix

7.1 Github Repo Link

<https://github.com/saideepreddy91/Fashion-MNIST>

7.2 Github Repo Readme

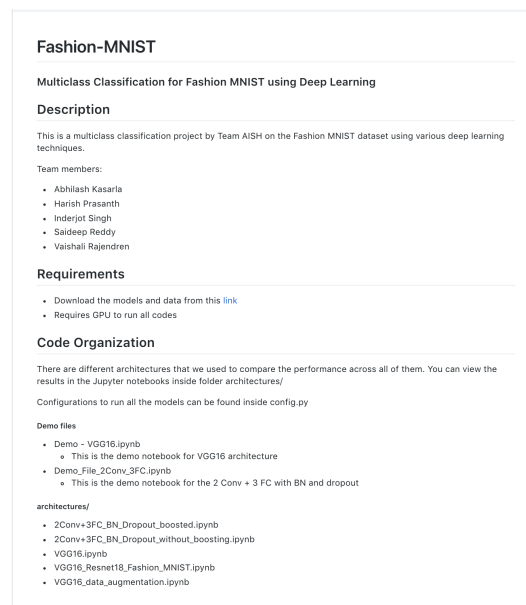


Figure 7.2 ReadMe

8 Bibliography

- Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms - <https://arxiv.org/abs/1708.07747>
- DropPruning for Model Compression - <https://arxiv.org/pdf/1812.02035.pdf>
- ResNet - <https://arxiv.org/pdf/1512.03385.pdf>
- VGG - <https://arxiv.org/pdf/1409.1556.pdf>
- Gradient Based Learning Applied to Document Recognition - LeNet