Influence of Regularization

I. Methods

Dataset Information:

The dataset “Fashion MNIST” was used to conduct the experiments. It’s a dataset of Zalando’s article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Every example is a 28x28 grayscale image with the labels for 10 different classes of different clothing categories. The model was downloaded directly from the TensorFlow library but the detailed description of the dataset is available on GitHub (https://github.com/zalandoresearch/fashion-mnist).

The labels include the following items:

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

Dataset Splitting:

The model was already splitted into training and testing set in the ratio: 85/15. To separate the validation set from the training data, I used the validation_split function and executed it on the training data to hold another 15% from the 85% of the training data and the ratio of the split in the end is: 70/15/15 for all the experiments.

Neural Network Architecture:

The hyperparameters for the model were selected once and set for all the experiments including the top-performing final model.

The hyperparameters used are:

```
num_iter = 2000
opt = 'adam'
num_filters = 32
kernel_size=3
pool_size = (2,2)
strides= (2,2)
activation = 'relu'
padding = 'SAME'
loss = "categorical_crossentropy"
epochs = 10
batch_size = 128
```

Number of iterations chosen for all the models was 2000. The optimization techniques used for the model was Adam. The results of the models optimized with Adam techniques are generally better than other chosen algorithms. It's known also for faster computation and requires fewer parameters for tuning. In machine learning models, it is a well known technique and it's often set as a default optimizer. Number of filters was chosen to be 32. Since I dealt with the image classification problem, there are a lot of features to learn by the model. The bigger number of filters lets the model capture more detail from the first more general ones to the more detailed and subtle features like the actual textures of the clothes, compared to the general features of the picture like colors and edges in the first layers. The more complex pictures get, the larger combinations of patterns are there to capture. The size of the filters, so the kernel size, was set to 3 to be able to move through the filters with the width and height of 3. Every pooling layer used in the experiments had a size of 2x2 and all the layers were strided by 2x2 layers. It means that the layer uses a filter of 2x2 and slides it over the entire image. Every time it stops (by 2 'spots'), the filter makes the calculations of the matrix. The activation function used for all the layers (except the final layer where 'softmax' is used for the output calculations) was 'relu' function. I chose the non-linear function that is a pretty popular choice in ML architectures. One of the biggest advantages of using the ReLU function is that it does not activate all the

units in the layers at the same time so it's computationally more efficient. The neurons are activated only if the output is more than 0. This way the model also saves time. The padding technique was not used in the model architecture. Padding parameter is set to 'SAME' which means that the filter is applied to all the elements of the input. It results in more time to train the model but the image examples in inputs get fully covered by the filters in the stride 2x2. I chose the categorical cross-entropy function as the loss function because it's a pretty common choice for multi-classification problems. The success of this loss function for the multi-classification process lies in the fact that one example can be only considered to belong to one of the 10 categories labeled in the model and should assign the probability of 0 to all the other classes. Number of epochs was set to 10 which means that the model completes 10 passes through the training set. The batch size used for all the experiments was set to 128. It means that the number of samples processed before the model updates is 128. Choosing the batch size that is power of 2 helps GPU Amake the process smoother and faster because of the way memory is set up in GPUs in general.

In the experiment phase, there were 8 distinct models trained on the same training set (70%) and tested on the validation set extracted from the validation_split function available in Keras module. The models are set up in the following manner:

Model 1: no regularization / 3 conv layers
Model 2: no regularization / 1 conv layer
Model 3: L2 regularization / 3 conv layers
Model 4: L2 regularization / 1 conv layer
Model 5: Dropout / 3 conv layers
Model 6: Dropout / 1 conv layer
Model 7: Early Stopping / 3 conv layers
Model 8: Early Stopping / 1 conv layer

The purpose of the experiments was to check how the number of convolutional layers affect the performance of the model and how different regularization techniques do on different models. One thing I was very interested in was how the models with only one convolutional layer predict the results. I did not set up the high number of convolutional layers to capture if increasing the number of convolutional layers of only 2 will have an impact on the performance. The experiments with the regularizations were set up to first check if the model without any regularization can have a similar performance to the ones where regularization was used. The first 2 models were treated like a sort of control group to all the other experimental groups with regularization techniques.

Hardware Information:

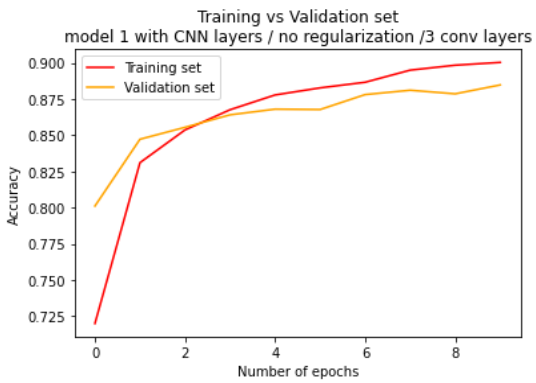
Processor: AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx 2.00 GHz
Installed RAM: 8.00 GB (6.96 GB usable)
System Type: 64-bit operating system, x64-based processor

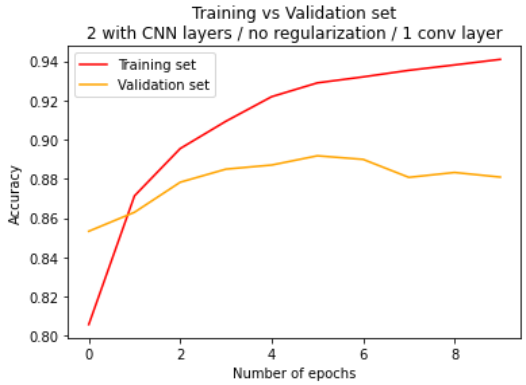
II. Results

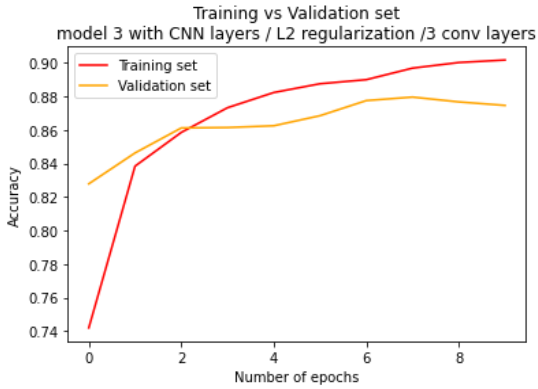
Results of Experiments:

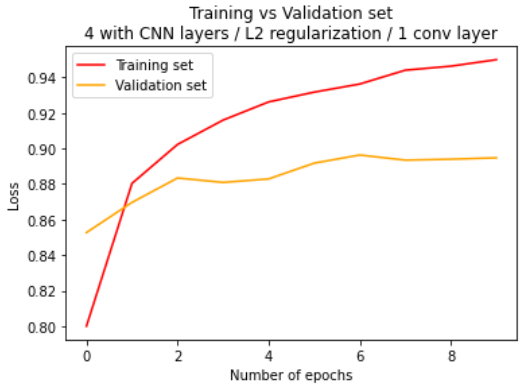
	# conv layers	Regularization	Train time (min)	Val accuracy	Test accuracy	Test loss
Model 1	3	-	5.03	0.885	-	-
Model 2	1	-	4.4	0.881	-	-
Model 3	3	L2	5.4	0.875	-	-
Model 4	1	L2	4.4	0.895	-	-
Model 5	3	Dropout	4.8	0.886	-	-
Model 6	1	Dropout	4.4	0.885	-	-
Model 7	3	Early Stopping	4.4	0.878	-	-
Model 8	1	Early Stopping	3.4	0.879	-	-
Top_model	1	L2	5.4	-	0.95	0.39

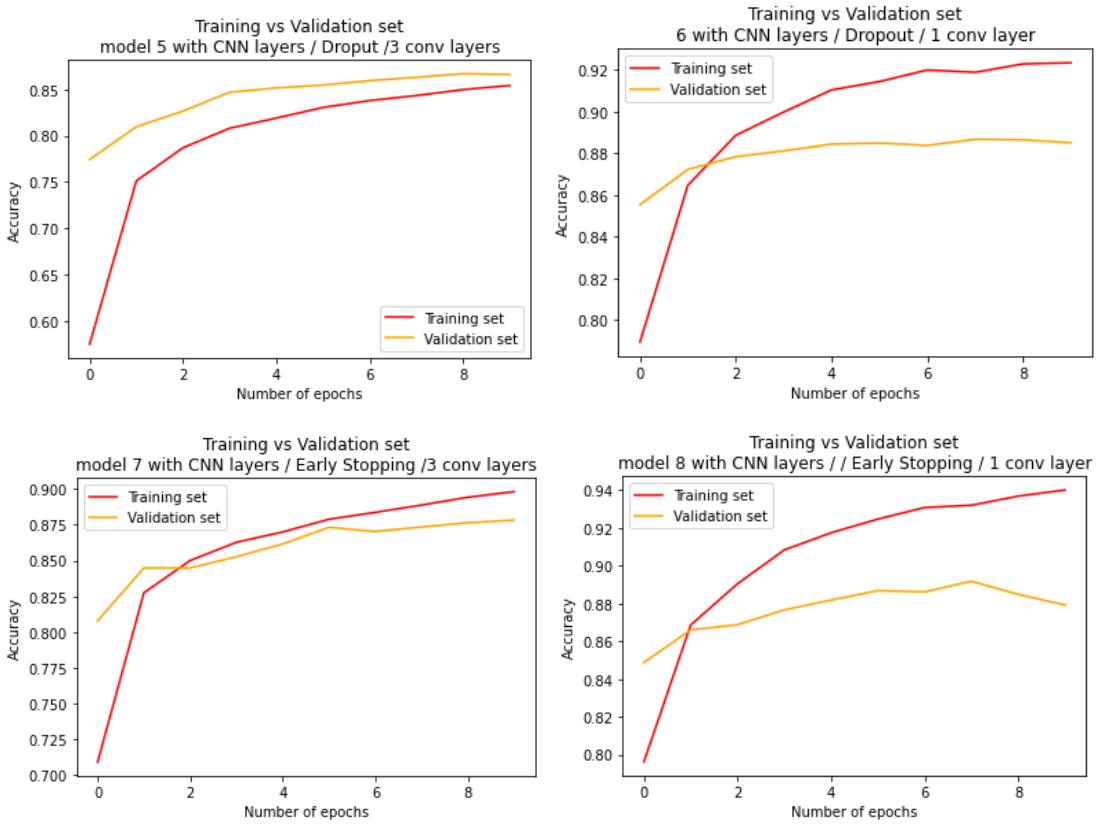
Learning Curves:











III. Analysis

Trends Observed:

One of the trends that was surprising is the fact that the number of convolutional layers does not consistently improve the performance of the models. In the case with no regularization, the model improves but only by 0.004 in the accuracy by adding 2 more convolutional layers. However, in the case on model 3 and 4 where L2 was used, the model with 1 convolutional layer did better than the one with 3 convolutional layers. The model 4 did the best overall on the validation test and is the top-performing model to be used later in the experiments. When comparing model 5 and 6 (dropout), the accuracy was only 0.001 percent higher by adding 2 more convolutional layers. Though, in the case of the model 7 and 8 (early stopping), adding 2 convolutional layers decreased the performance by 0.001. Perhaps, the difference between convolutional layers number of 3 and 1 is not enough to give the model a significant improvement (over 0.5%). I suspect that additional experiments with the bigger number of convolutional layers (5-10) would increase the model performance. The difference between model 4 and 5 is 0.02 which is the biggest difference when comparing the variations in the number of convolutional layers. Surprisingly, the model with 1 convolutional layer did better.

Another trend that can be observed from the results is that in all cases the validation accuracy was in the end smaller than the training accuracy, except one case, Model #5 with dropout and 3 convolutional layers did better on the validation set. This

happened only once in all the experiment and it might be caused by the fact that the dropout technique was used. The model at validation time is more robust because during the training process a part of the training set is dropped out. Additionally, dropout gives more generalization to the model but it can also be an imbalance of the data. One of the possible reasons why this model has a higher accuracy on the validation set is also that the dataset split was not done randomly. And hence the validation set may be completely different, it might have less noise or less variance from the training set. This may lead to the fact that it's actually easier to predict the accuracy. Additionally, if the validation set is small compared to the training set, the model will fit better the validation set rather than the training set. It refers back to the fact that the model used dropout if shrunk the number of examples in the data. Another conclusion from this observation is that dropout penalizes model variance by randomly freezing neurons in a layer during model training. Unfortunately, this might also mean that the dropout causes model underfitting and more data is needed to perform the experiment in the correct manner.

In the cases of models: 2, 3 and 8, the accuracy curves start going lower on the validation set at the end of the training. It might be the sign that the model starts overfitting the data. In model 1 there was no regularization technique used and I suspect this is the reason for the training curve to be smooth and performing well while the validation curve starts going down. The model is not robust enough because it also has only 1 convolutional layer. The fact that there was no regularization used, does not help to balance and normalize the data so it learns successfully the examples in the training data and it fails on the unseen examples from the validation set. Since the number of convolutional layers is low, the model doesn't have time and enough space to learn complex features and generalize well. Additionally, since the data might be noisy, it's learning the noise and starts predicting it well in the train set. Again, when predicting on never seen new examples, it fails.

Another trend I can observe from the learning curves is that the difference between the validation and training accuracy is smaller for all the models with 3 convolutional layers versus the models with 1 convolutional layer. This observation proves that more experiments with a higher number of convolutional layers may improve the general performance. Adding more layers helps extract more features and since I dealt with image classification, pictures require a robust number of features to be learnt to predict the image successfully. However, adding the layers to the architecture would increase the time needed for training and computational resources necessary to train the model. There is a trade off between the size of the architecture and the time and computational efficiency that would need to be considered.

The training time observed in the experiments was not very surprising. Model 1 took the longest ro train. This model did not use any regularization and used 3 convolutional layers. It confirms that the size of the model influences the training time, the bigger the model, the longer it trains. The lack of regularization techniques made it more difficult for the model to extract the features. Model 8 took the least amount of time. It used early stopping and only 1 convolutional layer. The use of early stopping can be the reason for a shorter training time since it stops the model at some point when performance starts decreasing. The overall trend observed from the training time is that

the bigger model size results in the longer training time. Every model configuration, with respect to the regularization usage, had a shorter training time when the architecture included 1 convolutional layer, instead of 3.

The top-performing model has an increase of the training time (1 minute increase) while training on the full training + validation sets and test on the 15% of the test set. The accuracy on the test set compared to the accuracy of the validation set was also significantly higher (0.895 vs 0.95 on validation and test set respectively). One of the reasons can be extending the number of training examples by adding the validation set to the training process in the top-performing model and testing on a small 15% only, compared to 85% of training and validation set. The model can generalize better when more data is fed to the model so it learns the features better. This fact could also help in balancing the data. By adding more images, the model learnt more diverse representations of different classes and generalized better. The validation set was used only to tune parameters.

Interpreting CNN Representations

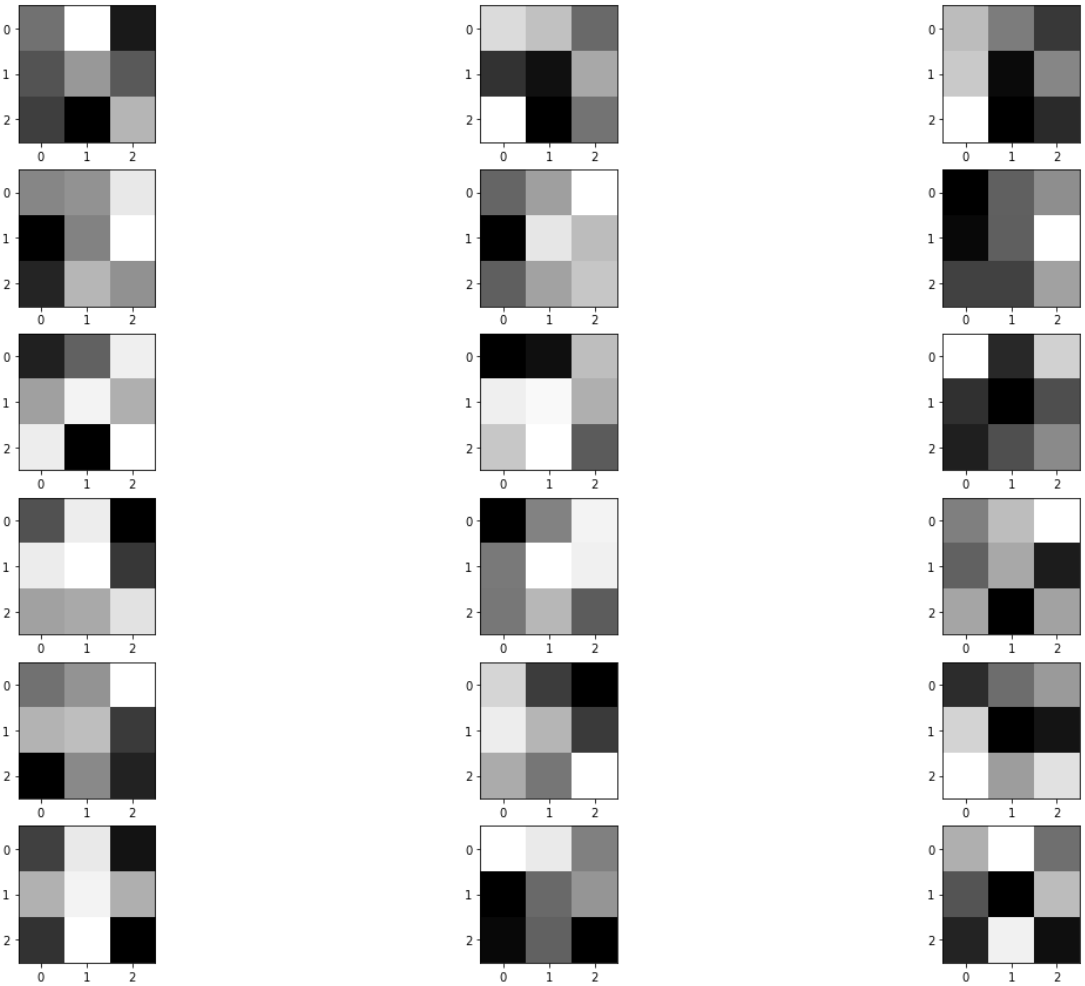
I. Methods

Model Architecture:

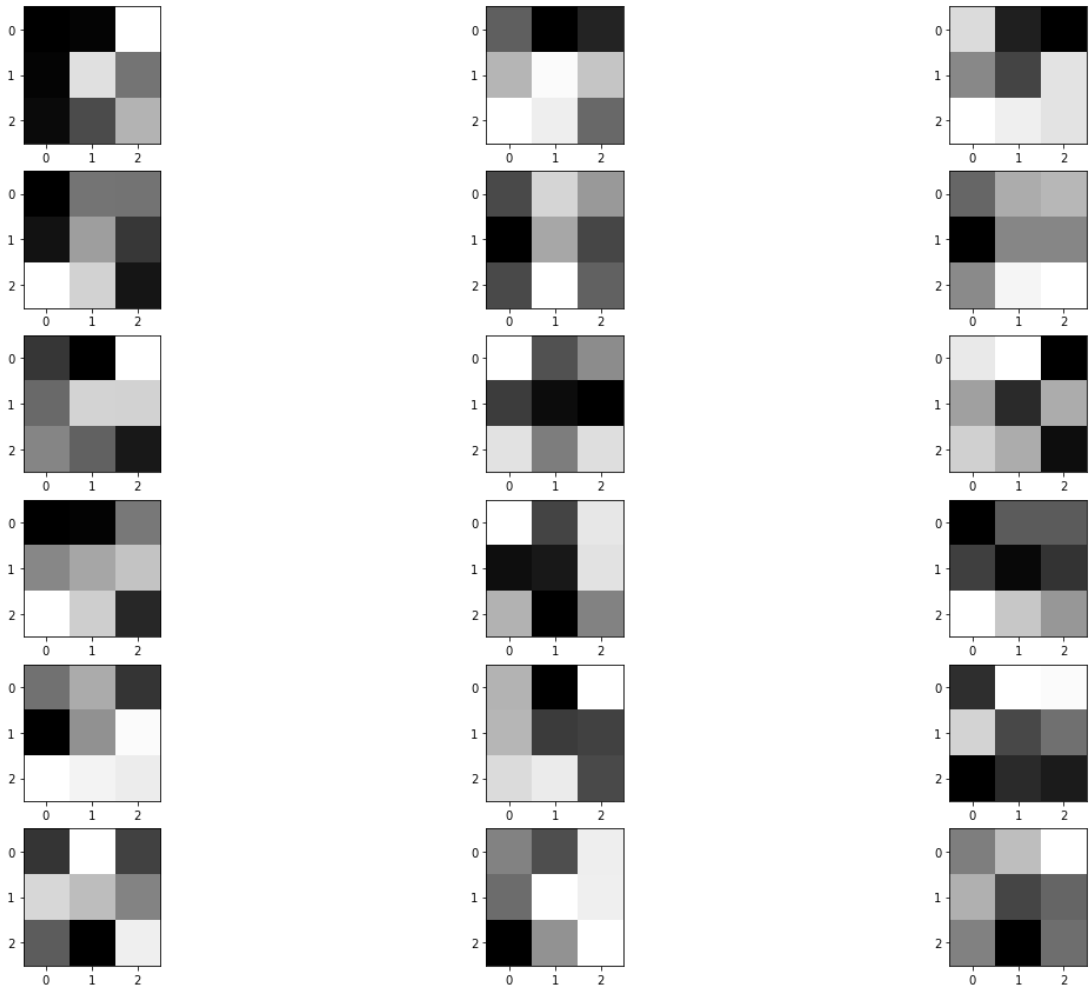
Since the best performing model from the previous 8 experiments had 1 convolutional layer, I built a new model with extra convolutional layers to be able to compare the results from multiple layers. The visualization model includes 4 convolutional layers and this is the only difference between the top-performing model from the previous part of the research. 3 more convolutional layers were added. All other hyperparameters selected and features of the model stayed the same.

II. Results

Visualization of 1st Convolutional Layer:



Visualization of 4th Convolutional Layer:



III. Analysis

Trends Observed:

First observed trend from the visualization of the convolutional layers is that the first channels of both layers are more concentrated at the edges of the image and as we're getting to the second and third channel, the point of focus is switched to the center of the image. This might mean that the channels subsequently over-first feature the pictures at the edges to detect the general trends of the image and then focus on more details that are mostly at the center of an image.

The second trend that can be observed is comparing the first and the fourth convolutional layer. It's clear that the dark squares on the fourth convolutional layer are more concentrated together. It means that the model in the further layers is looking for more condensed patterns. For example, 3 squares next to each other have a similar color on the greyscale. It means that this space on the picture is forming some kind of a

pattern, maybe a single line or it's a part of a bigger object. In the first layer of convolution the model recognizes these parts of one object only separately but does not put them together into bigger pictures.

The overall conclusion is that the model is not able to capture more complex features at the beginning of the training and each layer is responsible for some kind of different feature extraction. The deeper the model gets, the more complex features are recognized. This is why the visualizations show more condensed and more grouped together patterns on the further layers.