An InceptionResnet based Image Classifier for Fashion-MNIST Dataset

Ma Jiawei
jiawei.ma@studenti.unipd.it

1. Introduction

In this report we describe some *image classification* experiments performed on the Fashion-MNIST dataset using machine learning techniques. Our main focus were on the deep learning tools such as Convolutional Neural networks(CNNs), in particular, we were inspired by the neural network architecture based on *InceptionResNet* and we set up only one Inception block with residual connection since we were dealing with low resolution and gray scale images. This *InceptionResNetFashion* model gave us quite satisfactory results with *low missclassification error* and no overfitting behaviour was observed, the model was medium-sized and there is still room for improvement.

2. Dataset

Our data come from the novel image dataset *Fashion-MNIST*, mainly used for benchmarking machine learning algorithms. Each data is a 28×28 gray scale image, associated with a label from *10 classes*. More details about this dataset (how and why is build) are well-illustrated in [3], here we want to stress that this new dataset aims to replace the old MNIST dataset which is *too easy and overused* (classic ML algorithms can also achieve 97 percent of accuracy).
In our tests we rearranged this dataset in the following way:

- **Training set:** it consists of 5000 samples of shape 28×28 along with the respective labels.
- **Validation set:** it includes 1000 images and labels with same shape to the training set.
- **Test set:** it's composed of other 1000 images with *no labels* and the purpose of this set of images is to participate to an open Kaggle Competition program[1].

Most importantly, we verified that the training and validation dataset are *balanced between classes*, that is 10 percent for each class; we believe that this is the first analysis step in any machine learning experiment otherwise the returned results will be unreliable.

2.1. Data preprocessing and augmentation:

we visualized some data and it turned out that the images belong to same class are well captured and positioned (for instance, no significant rotations are observed). Thus, the only reasonable data augmentation operation, during the training phase of the models, was to randomly flip the images in *horizontal direction.* As preprocessing operation we *normalized* (dividing each pixel value by 255, the default maximum value) the input data in order to accelerate the training and to have best model performance, this is what proved in [1].

3. Description of models and method

Since CNNs are noted by their great performance in image classification problems we preferred this approach but adding some other advanced features to the simple CNNs.
The most important modifications carried out are *Inception* architecture (very deep CNNs) with *residual* connection. The core idea of this architecture is to *concatenate* many CNNs with different characteristics/hyperparameter (filter size, number of channels and strides) and this leads to a better model capable to learn more feature from the data.
In [2] it's proved that Inception network with residual blocks presented clear acceleration to the training phase and there is also some evidence of residual Inception networks outperforming similarly expensive Inception networks without residual connections by a thin margin.
Nevertheless, in this work we propose a 'smaller' version of *InceptionResNetV4* illustrated in the paper and we call it *InceptionResNetFashion* model, this reduction of size is due to our input data which have low resolution with respect to what the authors of the InceptionResNet model had experimented with.

3.1. InceptionResNetFashion model

The very basic element of the model is the *2D convolutional layer with batch normalization* (conv2d-bn). The benefits from batch normalization gates are well explained in [1]. We have four main elements: *Input block, Inception-*

[1] https://www.kaggle.com/c/image-classification-fashion-mnist

ResNet block, Reduction block and Output block.
To be brief,

- **Input block** is a conv2d-bn layer with 3×3 filter size and it is meant to be a first feature extraction block.
- The aim of having **Inception ResNet block** is the same; maximizing as much as possible the power of feature extraction (stacking convolutional batch normalization layer and concatenating created channels).
- We use **Reduction block** to decrease the input data dimensions (incresing the filter dimensions).
- Finally, **Output block** is useful to reduce, once more, the dimensionality and from here the data are ready to go through a fully connected layer (dense layer) which makes predictions based on extracted information.

We want to mention that we added more *dropout layers* than how many the original model has and they are served to prevent overfitting phenomena, although the Inception net generally has great generalization ability (the model's overview is in fig. 3 in Appendix).

3.2. Method

3.2.1 grid search hyperparameters

First of all we *grid search hyperparameters* for our model and the hyperparameters are:

- **Length**: It is the number of InceptionResnet blocks we stack up. It's interesting to investigate if, in this case, the size of the network could be relevant, because large neural network may lead to *overfitting*.
- **Width**: It measures how wide is the Inception block; it's the number of different CNNs *branches* we concatenate.
- **Dropout rate**: it's always suggested to use dropout layers to avoid the overfitting and the right rate depends on the data.

One may believe that more wider and/or deeper is the network better results will be obtained, but in our opinion, there is a *limit size* and bigger model won't have improvements, it also requires more training steps and computational resources.
Finally, we tried to understand which hyperparameter adjustment gave the most relevant model improvement. Then we select the best one according to many evaluation metrics and we trained again the selected model to achieve the optimum values.

Hyperparams	**train-acc**	**val-acc**	**T. params**
L=3, W=4, D=0.5	0.911 $(-,-)$	0.912	$453 \cdot 10^3$
L=4, W=4, D=0.5	0.905 $(-0.006,-)$	0.908	$564 \cdot 10^3$
L=3, W=5, D=0.5	0.906 $(-,-0.005)$	0.914	$861 \cdot 10^3$
L=4, W=5, D=0.5	0.908 $(0.002,0.003)$	0.912	$1100 \cdot 10^3$
L=3, W=4, D=0.3	0.925 $(-,-)$	0.924	$453 \cdot 10^3$
L=4, W=4, D=0.3	0.928 $(0.003,-)$	0.93	$564 \cdot 10^3$
L=3, W=5, D=0.3	0.931 $(-,0.006)$	0.932	$861 \cdot 10^3$
L=4, W=5, D=0.3	**0.935 (0.004,0.007)**	**0.934**	**1.1M**

Table 1. Grid Search Hyperparameters (12 epochs). Bold values are related to the best model. The numbers below training accuracy values are the gains by increasing length or width and fixing dropout rate: (L, W).

3.2.2 Training method

Our model's output is a standard image classification model's output, *softmax* output, that is a simple layer with number of classes units, each enumerated unit represents the probability that the correspondent class is the correct/true class. So the loss function is *multi-class cross-entropy*:

$$\mathit{CE} = -\sum_{i=1}^{D}\sum_{j=1}^{C} t_{ij}\log(s_{ij}),$$

where C is the number of classes, D is the size of training data, t_{ij} are binary indicators (0 or 1) if class label j is the correct classification for observation i and s_{ij} are the values of the softmax function. For this optimization problem *Adam Optimization Algorithm* with standard parameters is preferred since it generally has better results applied to big neural networks. One crucial aspect is that we reduced learning rate during the last training steps when the loss got stuck; this is quite common practice to help the algorithm to converge.

4. Experiments

Now we illustrate the results of our experiments. In assessing the model we also relied on another two very important metrics in the classification tasks: *AUROC* measure and on *confusion matrix* in order to avoid a *biased* model, they are often ignored and to the accuracy score is frequently given too much weight. In this multi-class classification problem we preferred to calculate the mean of AU-

train-acc	train-loss	val-acc	mean-AUROC
0.992	0.0563	0.94	0.997

Table 2. Best model performance.

ROC measures computed in the *one-v.s.-rest* manner.
More detailed graphic information are contained in attached files[2].

4.1. results from Grid Search

We discovered from the Grid Search that there are no significant differences between these models, both increasing the length and increasing the width returned better results, the width seemed to affect more the model performance. See the Tab. 1. We note that higher dropout rate is not suitable and this may due to the dataset analysed as just raised; the images are not affected by noises. Thus, the the best model, in term of *accuracy on training and validation dataset*, is the largest one and we can draw also two important insights about the model:

- on one hand the biggest model required more time for each training step, on the other hand it had more remarkable precision gain after each training cycle;

- it was clear that all models tested generalized well the data because the misclassification error gaps, between train and validation data, was very small and proves that the Inception block could indeed extract useful features from the data.

We saw that, as desired, the Inception architecture was able to extract correct features avoiding overfitting, we can verify it observing the small gap between the accuracy on training data and the accuracy on validation data. Furthermore, the residual connection could indeed accelerate the training phase.

4.2. Training the best model and results

Once computed the parameters we restarted the training step and the final trained model was obtained after 40 *epochs*, the training data recorded are reported in fig. 1. The results are *very satisfactory*; for sure our model *overperformed* traditional machine learning techniques (many experiment results are reported in [3] and it's better than the SVM model we built) and it seems to be more precise (accuracy) also than other deep learning based models [3].
Thanks to the residual connection the highest validation data accuracy and lowest validation data loss were achieved after 18 epochs, the highest training data accuracy is larger

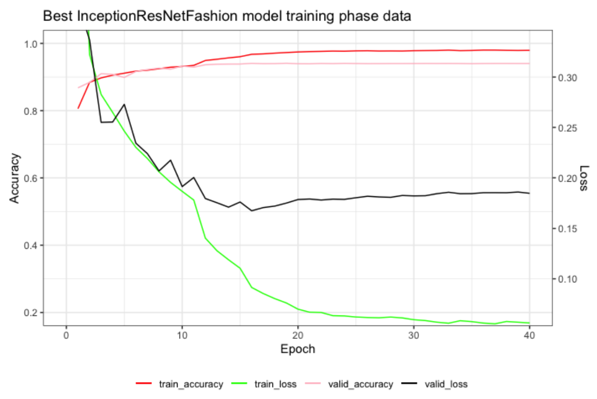


Figure 1.

than the final accuracy on validation data and this probably means that the training set doesn't contain all information about other possible images. We had a very good AUROC score and looking at the confusion matrix (fig. 2 in Appendix) we realized that the misclassification error was mainly due to a specific class (*Shirt* class). Fortunately, the model had no bias problem.
Ideally, according to what we see from the results we can reach even more training accuracy or lower training loss by repeating training steps, neverthless, the validation accuracy and loss are saturated as noted before.

5. Conclusion

We experimented with Inception architecture using the novel Fashion-MNIST dataset, we verified the model's feature extraction ability as illustrated in the reference paper and the training phase acceleration by leveraging the residual connection.
Our basic model performed well with respect to many machine learning techniques but there are also several more precise deep learning models. Although, we should take into account also the model size (measured by the number of trainable parameters) which represents a bottleneck in many situations and the neural network we built is of medium size. Certainly, we have still room for improvement by making wider or deeper neural network but there is always a trade-off between accuracy and speed.

References

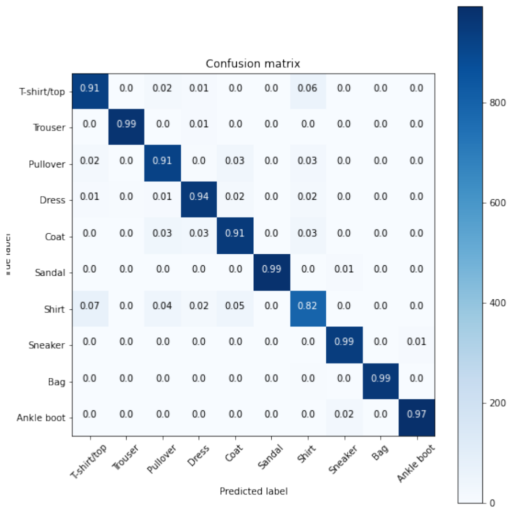
[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[2] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[3] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[2]The training logs are provided in Tensorboard format.
[3]https://github.com/zalandoresearch/fashion-mnist.
The evaluation method is unknown.

Appendix



	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	0.91	0.0	0.02	0.01	0.0	0.0	0.06	0.0	0.0	0.0
Trouser	0.0	0.99	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0
Pullover	0.02	0.0	0.91	0.0	0.03	0.0	0.03	0.0	0.0	0.0
Dress	0.01	0.0	0.01	0.94	0.02	0.0	0.02	0.0	0.0	0.0
Coat	0.0	0.0	0.03	0.03	0.91	0.0	0.03	0.0	0.0	0.0
Sandal	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.01	0.0	0.0
Shirt	0.07	0.0	0.04	0.02	0.05	0.0	0.82	0.0	0.0	0.0
Sneaker	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.01
Bag	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.0
Ankle boot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.0	0.97

Figure 2. Final Confusion Matrix

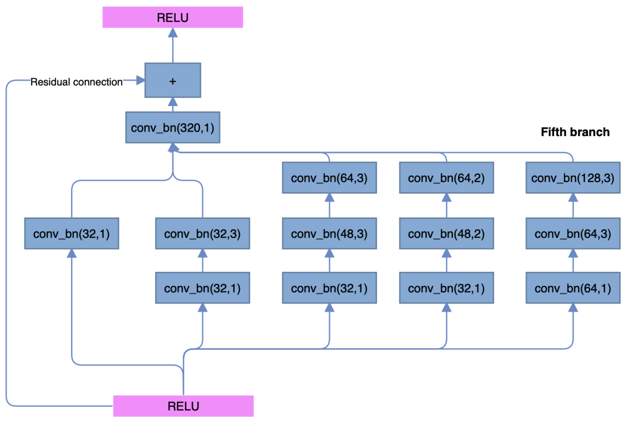


Figure 4. InceptionResenet block.

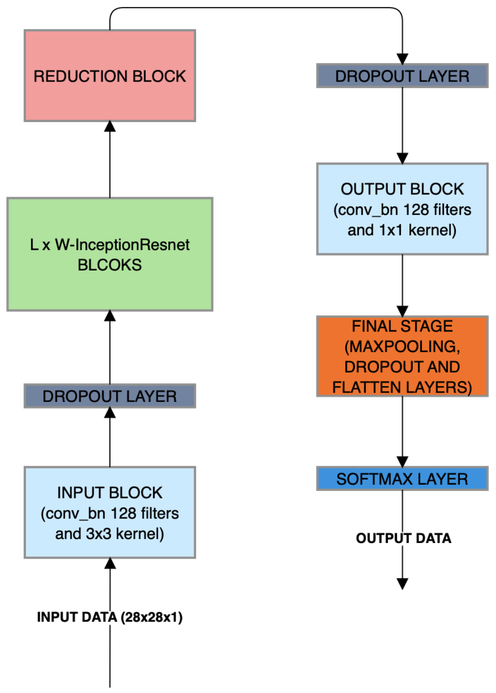

REDUCTION BLOCK
L x W-InceptionResnet BLCOKS
DROPOUT LAYER
INPUT BLOCK (conv_bn 128 filters and 3x3 kernel)
INPUT DATA (28x28x1)
DROPOUT LAYER
OUTPUT BLOCK (conv_bn 128 filters and 1x1 kernel)
FINAL STAGE (MAXPOOLING, DROPOUT AND FLATTEN LAYERS)
SOFTMAX LAYER
OUTPUT DATA

Figure 3. Model structure.