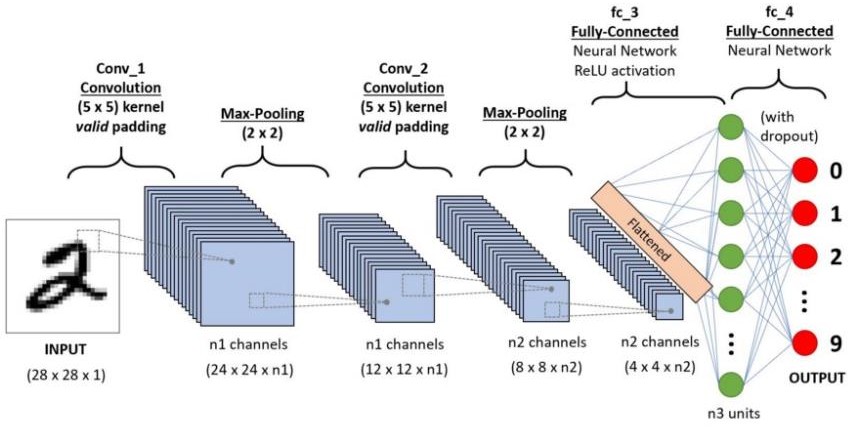
Proceedings of Knowledge Discovery Course -UNICT1[–](#_bookmark0)3, 2022 Report for Homework 1

## Fashion MNIST Classification using CNNs

**Muhammad Hassan Ali** [LAIMMM93H22Z236F@STUDIUM.UNICT.IT](mailto:laimmm93h22z236f@studium.unict.it)

# Introduction to Convolutional Neural Networks (CNN):

Convolutions are a way to capture information about the ordering of pixels. The type of convolutions are interested in are 2d discrete convolutions, which act like a weighted sliding sum over an area of pixels. For instance, a 3x3 matrix called a kernel slides across the pixels in an image. At each point, it calculates the weighted sum of the kernels’ values and every pixel in the 3x3 chunk of the image. The sum is then put in the first value of the output image. The kernel then slides over one pixel and repeats the process for every pixel in the image.



In [deep learning,](https://en.wikipedia.org/wiki/Deep_learning) a convolutional neural network (CNN) is a class of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analyze visual imagery. Now when we think of a neural network, we think about matrix multiplications but that is not the case with ConvNet.

Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc**.**

# Dataset:

Fashion-MNIST is a dataset of [Zalando](https://jobs.zalando.com/tech/)'s article images. Fashion training set consists of 70,000 images divided into 60,000 training and 10,000 testing samples. Dataset sample consists of 28x28 grayscale image, associated with a label from 10 classes.

The 10 classes are as follows:

{T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot}

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 785 columns. The first column consists of the class labels (see above) and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

* To locate a pixel on the image, suppose that we have decomposed x as x = i \* 28 + j, where i and j are integers between 0 and 27. The pixel is located on row i and column j of a 28 x 28 matrix.
* For example, pixel31 indicates the pixel that is in the fourth column from the left, and the second row from the top, as in the ascii-diagram below.

## Labels:

Each training and test example is assigned to one of the following labels:

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

**Building a CNN:**

Our Neural Net has following layers:

Two Sequential layers each consists of following layers-

Convolution layer that has kernel size of 3 \* 3, padding = 1 in 1st layer and padding = 0 in second one. Stride of 1 in both layer.

Batch Normalization layer.

Acitvation function: ReLU.

Max Pooling layer with kernel size of 2 \* 2 and stride 2.

Flatten out the output for dense layer(a.k.a. fully connected layer).

3 Fully connected layer with different in/out features.

1 Dropout layer that has class probability p = 0.25.

All the functionaltiy is given in forward method that defines the forward pass of CNN.

Our input image is changing in a following way:

First Convulation layer : input: 28 \* 28 \* 3, output: 28 \* 28 \* 32

First Max Pooling layer : input: 28 \* 28 \* 32, output: 14 \* 14 \* 32

Second Conv layer : input : 14 \* 14 \* 32, output: 12 \* 12 \* 64

Second Max Pooling layer : 12 \* 12 \* 64, output: 6 \* 6 \* 64

Final fully connected layer has 10 output features for 10 types of clothes.

# Training Procedure:

For training our model, first we build CNN model with two convolutional layers and added three fully connected layers. In 1st convolutionally layer we set up our parameters’ values (in\_channels, out\_channels and kernel\_size/filter) then applies maxpooling, the Pooling layer is like the convolutional layer and responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data by reducing the dimensions. There are two types of pooling average pooling and max pooling. In our case we’ve applied Max Pooling. Similarly, we created 2nd convolutionally layer respectively. After that we passes the above created layers in feedforward function and return output as variable ‘x’. In next step we defined optimizer and loss (use a classification Cross-Entropy loss and SGD with momentum).

Optimizer describes how the model is updated based on the data it sees and its loss function. And loss function measures how accurate the model is during training. We trained the network by simply loop over our data iterator and feed the inputs to the network and optimized. We have

trained the network for 2 passes over the training dataset in order to check if the network has learnt anything at all. We have checked this by predicting the class label that the neural network outputs and checking it against the ground-truth. If the prediction is correct, we added the sample to the list of correct predictions. Now we have trained a neutral network, the next step is to get all the predictions of this model. And build a confusion matrix. A simple way of building a confusion matrix is to use sklearn.metrics.confusion\_matrix, therefore we simply call the confusion\_matrix function to build a confusion matrix and plot it by using library ‘matplotlib’.

# Experimental Results

Several steps were performed from constructing CNN model to evaluation our model with confusion matrix and we seemed that the desire accuracy began at epochs\_3 therefore, we stopped at epoch 10 and concluded that:

Model Accuracy Epoch 0--- 71.6 %

Epoch 1--- 72.02%

Epoch 2--- 74.3%

Epoch 3--- 75.1%

Epoch 4--- 82.4%

Epoch 5--- 87.6%

Epoch 6--- 89.10%

Epoch 7--- 93.5%

Epoch 8---94.4%

Epoch 9---95.5%

## Graphical user interface Description automatically generated with medium confidenceConfusion Matrix: