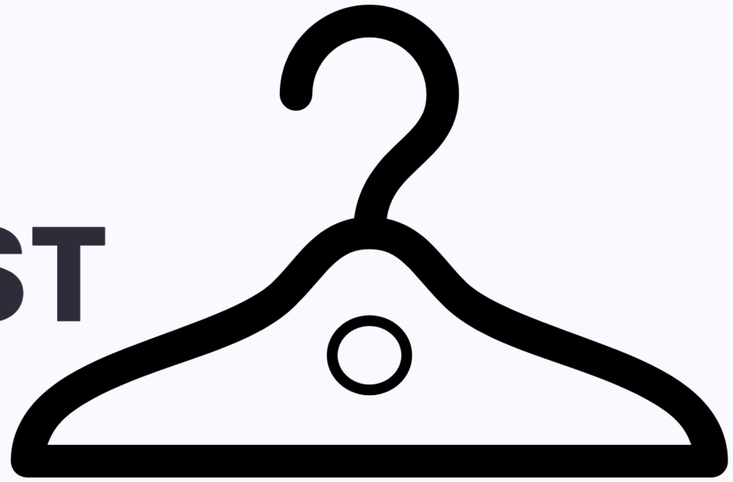
FASHION MNIST CLASSIFIER



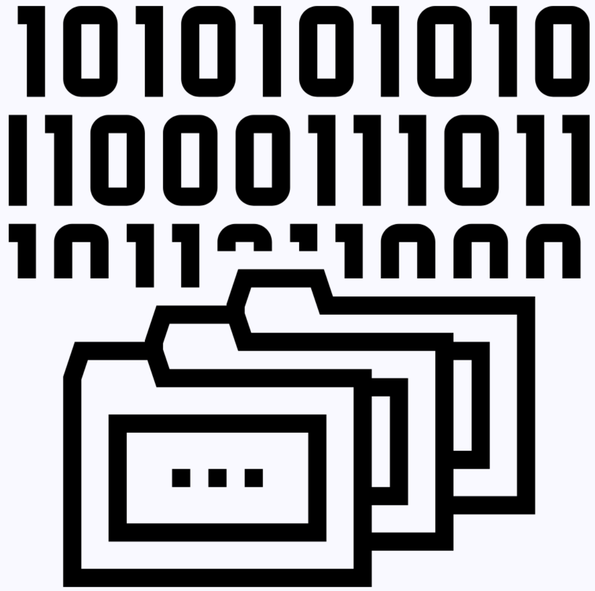
BY DANA DIAB

Table Of Contents

1. Data Preparation
2. Model Creation
3. Model Evaluation

01.

Data Preparation



Datasets

Each row is a different image representation in the form pixel data.

fashion-mnist_train.csv

```
train_df.head(10)
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...
0	2	0	0	0	0	0	0	0	0	0	...
1	9	0	0	0	0	0	0	0	0	0	...
2	6	0	0	0	0	0	0	0	5	0	...
3	0	0	0	0	1	2	0	0	0	0	...
4	3	0	0	0	0	0	0	0	0	0	...
5	4	0	0	0	5	4	5	5	3	5	...
6	4	0	0	0	0	0	0	0	0	0	...
7	5	0	0	0	0	0	0	0	0	0	...
8	4	0	0	0	0	0	0	3	2	0	...
9	8	0	0	0	0	0	0	0	0	0	...

10 rows × 785 columns

fashion-mnist_test.csv

```
test_df.head(10)
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...
0	0	0	0	0	0	0	0	0	9	8	...
1	1	0	0	0	0	0	0	0	0	0	...
2	2	0	0	0	0	0	0	14	53	99	...
3	2	0	0	0	0	0	0	0	0	0	...
4	3	0	0	0	0	0	0	0	0	0	...
5	2	0	0	0	0	0	44	105	44	10	...
6	8	0	0	0	0	0	0	0	0	0	...
7	6	0	0	0	0	0	0	0	1	0	...
8	5	0	0	0	0	0	0	0	0	0	...
9	0	0	0	0	0	0	0	0	0	0	...

10 rows × 785 columns

Conversion into numpy arrays of float32

```
array([[2., 0., 0., ..., 0., 0., 0.],
       [9., 0., 0., ..., 0., 0., 0.],
       [6., 0., 0., ..., 0., 0., 0.],
                    ...,
       [8., 0., 0., ..., 0., 0., 0.],
       [8., 0., 0., ..., 0., 0., 0.],
       [7., 0., 0., ..., 0., 0., 0.]],
      dtype=float32)
```

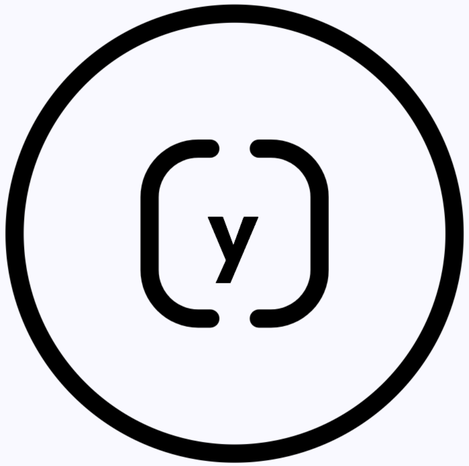
Train_data

This step is important to get the acceptable form for Tensorflow and Keras

Separation of X and y

with y being the first column of the dataset => dataset[: , 0] and X being the rest of the columns => dataset[: , 1 :]

T-shirt/top	0
Trouser	1
Pullover	2
Dress	3
Coat	4



5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Where X are image features and y are image label.
No null values were detected.

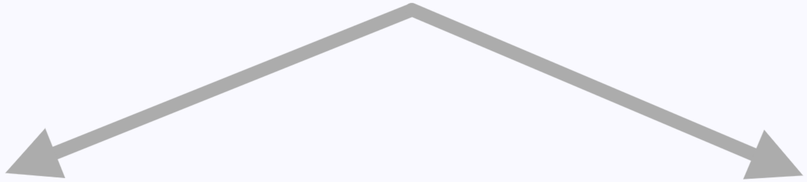
Normalization

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
                  ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]],
      dtype=float32)
```

X_train

Pixel values are normalized to the range [0,1]

Splitting training dataset



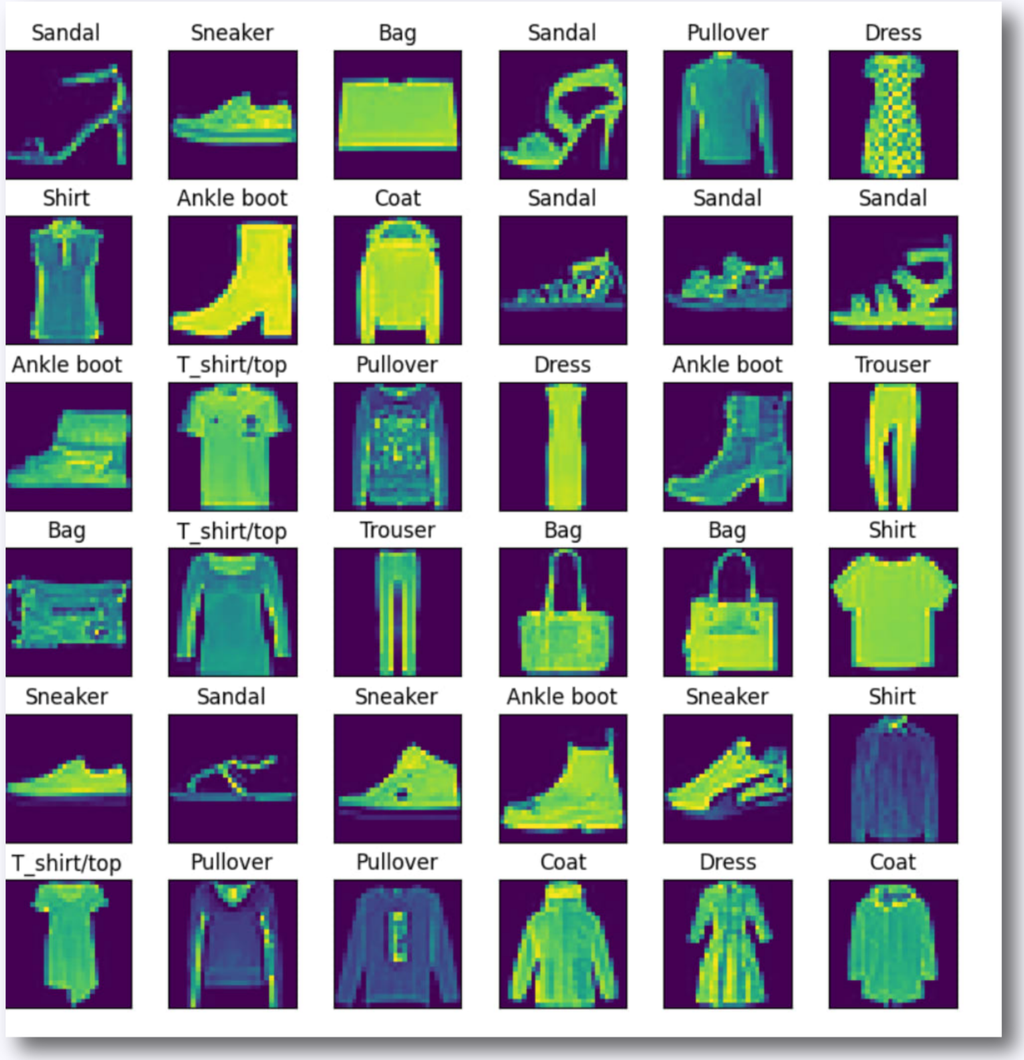
Training data

80%

Validation data

20%

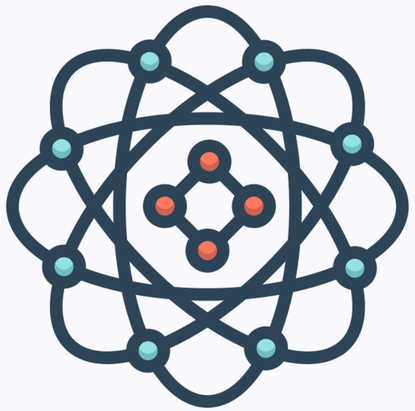
```
X_train, X_validate, y_train, y_validate = train_test_split(X_train,y_train,test_size = 0.2,random_state = 42)
```



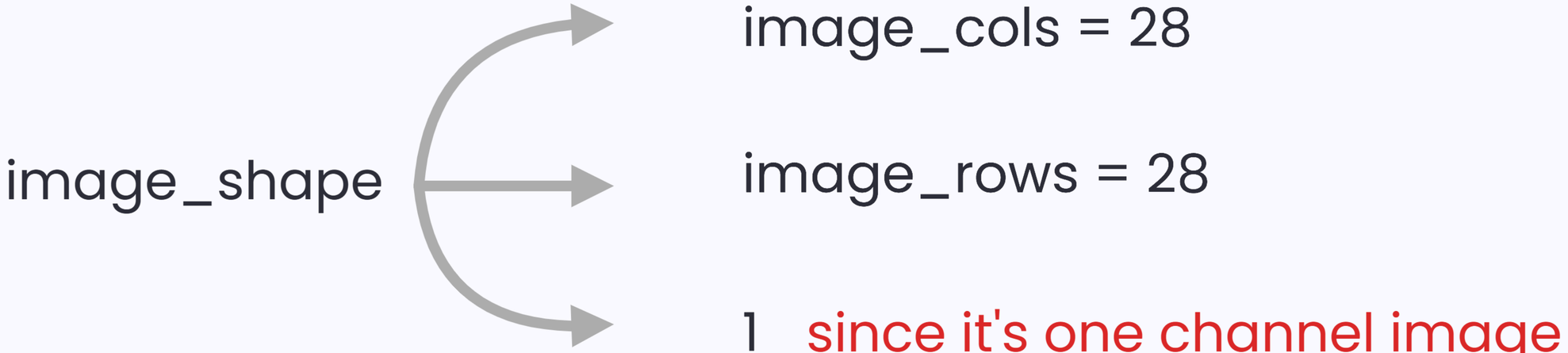
X_train dataset

02.

Model Creation



Preparing Image data



We tend to reshape the training, validation and testing dataset to provide an acceptable input format to the neurons.

Neural Network Architecture

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
dropout (Dropout)	(None, 13, 13, 32)	0
flatten (Flatten)	(None, 5408)	0
dense (Dense)	(None, 32)	173,088
dense_1 (Dense)	(None, 10)	330

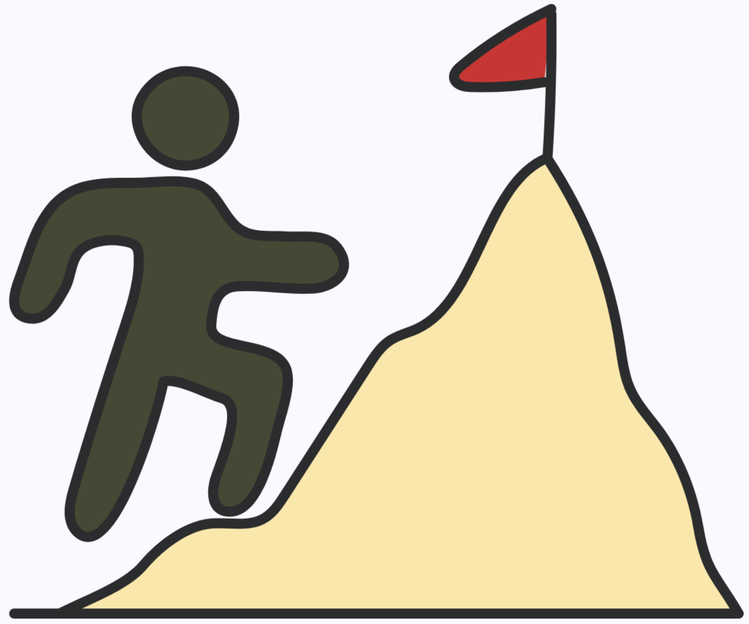
Total params: 173,738 (678.66 KB)
Trainable params: 173,738 (678.66 KB)
Non-trainable params: 0 (0.00 B)

03.

Model Evaluation

91%

accuracy



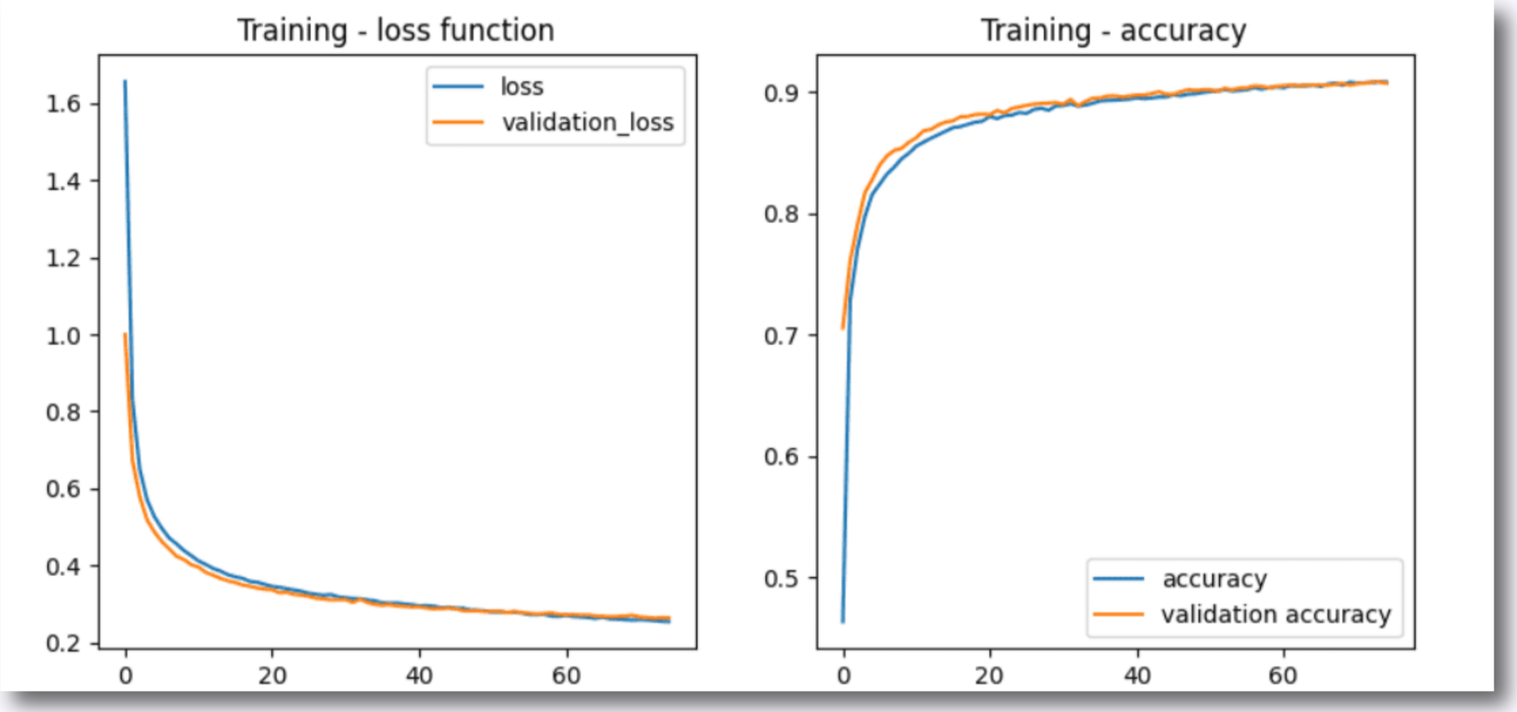
Classification Report

	precision	recall	f1-score	support
Class 0	0.86	0.86	0.86	1000
Class 1	0.99	0.98	0.98	1000
Class 2	0.89	0.84	0.86	1000
Class 3	0.89	0.94	0.92	1000
Class 4	0.86	0.89	0.87	1000
Class 5	0.99	0.97	0.98	1000
Class 6	0.76	0.73	0.75	1000
Class 7	0.95	0.97	0.96	1000
Class 8	0.97	0.98	0.98	1000
Class 9	0.96	0.97	0.96	1000
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000

What we notice:

- Underperformance for class 6 in terms of both precision and recall
- Slightly underperformance for class 2 in terms of recall

but....did we fall in the trap of Overfitting?



Loss and Validation loss both decrease

+

Accuracy and Validation accuracy both increase

which means...

NO OVERFITTING

Thank you

For my source code, feel free to click link below

=> https://github.com/Dan-data-tech/Fashion-MNIST.git