TP n°1 : Deep Learning

Image classification with Pytorch Lightning

Yohan ISMAEL & Théo BERILLON

Exploratory Data Analysis MNIST

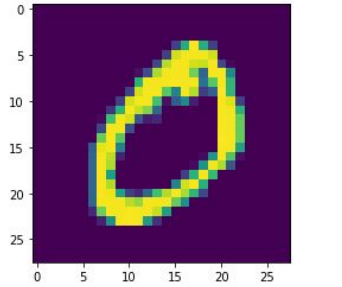
```
# TODO : Retrieve one sample of the Dataset.
sample = dataset_train[1]

# TODO : What is in a sample ? Print the sample to understand
print(sample)
```

```
(<PIL.Image.Image image mode=L size=28x28 at 0x7F54F3B3DE10>, 0)
```

```
plt.imshow(sample[0])
```

```
<matplotlib.image.AxesImage at 0x7f55
```



Training dataset composed of 60000 images of numbers between 0 and 9, and labels with the corresponding number (MNIST dataset).
Test dataset composed of 10000 images and labels.

Shape of image : 28x28x1 pixels in Grayscale

```
[ ] # TODO : What's the shape of the input image.
    shape = np.shape(sample[0])
    shape
```

```
(28, 28)
```

```
L = [0,0,0,0,0,0,0,0,0,0]
for i in range(len(dataset_train)):
  L[dataset_train[i][1]] += 1
print(L, "There is approximately the same number of occurences in each class, so the dataset is approximately balanced")
```

```
[5923, 6742, 5958, 6131, 5842, 5421, 5918, 6265, 5851, 5949] There is approximately the same number of occurences in each class, so the dataset is approximately balanced
```

Approximately balanced dataset

MNIST Classifier

The network:

	Name	Type	Params
0	layer_1	Conv2d	100
1	layer_2	Conv2d	91
2	layer_linear1	Linear	5.8 K
3	relu	ReLU	0
4	train_acc	Accuracy	0
5	valid_acc	Accuracy	0
6	test_acc	Accuracy	0

6.0 K Trainable params
0 Non-trainable params
6.0 K Total params

```
(layer1): Conv2d(1, 128, kernel_size=(3, 3), stride=(1, 1))
(relu): ReLU()
(layer2): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1))
(linear1): Linear(in_features=36864, out_features=10, bias=True)
(linear2): Linear(in_features=10, out_features=10, bias=True)
(sigmoid): Sigmoid()
```

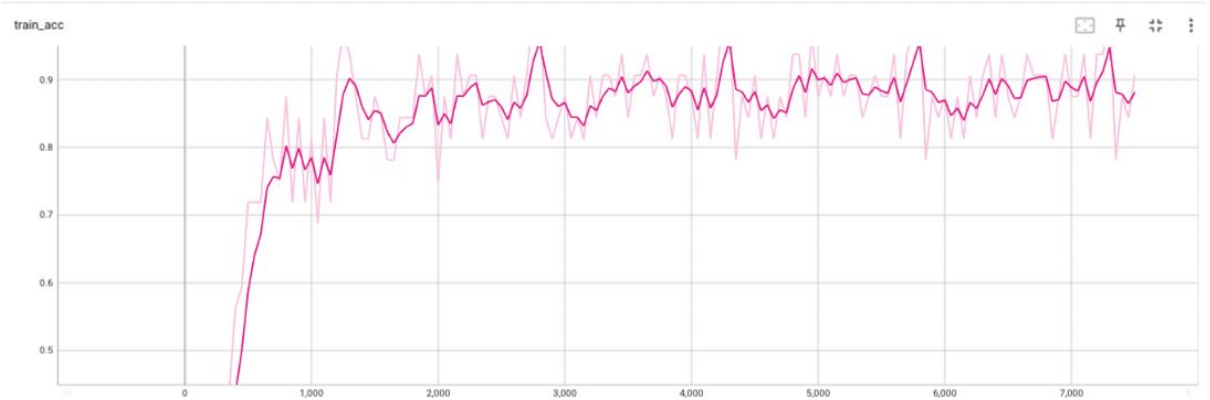
```
x=self.layer_1(x)
x=self.relu(x)
x=self.layer_2(x)
x=self.relu(x)
x = x.view(x.size(0),-1)
x=self.layer_linear1(x)
x=F.sigmoid(x)
return x
```

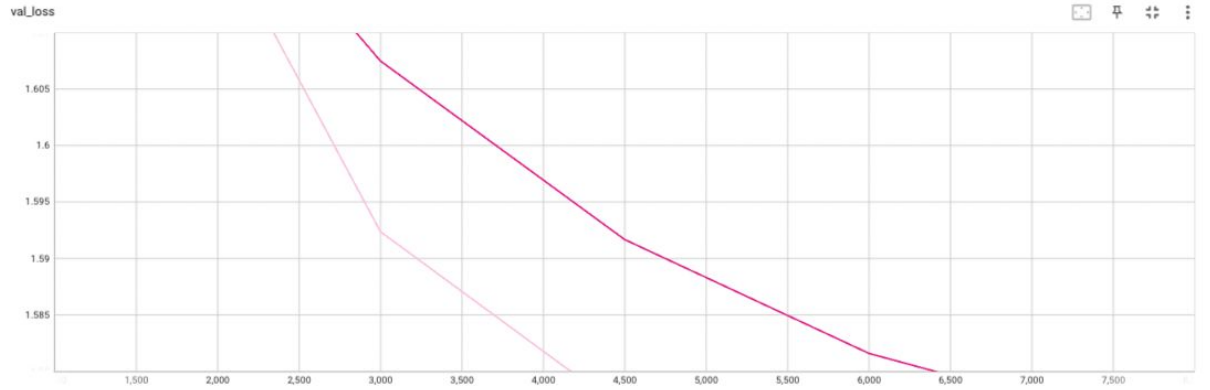
- The input channel of the first layer is 1 because we have Grayscale images.
- The number of output of the first linear layer is : (28-2)*(28-2)*64, 2 is the reduction of dimension due to kernel size (=3) after a Conv2d, 64 is the output channel of the previous layer.
- We use CrossEntropyLoss for classification with a learning rate of 0.01
- The forward method is the method we use to go through all the layers of the model.
- The training data will be used to train the model, the validation data will test the model at the end of each epoch and the testing data will allow to test the model after learning phase to see if it is able to recognize numbers on the images.

Results

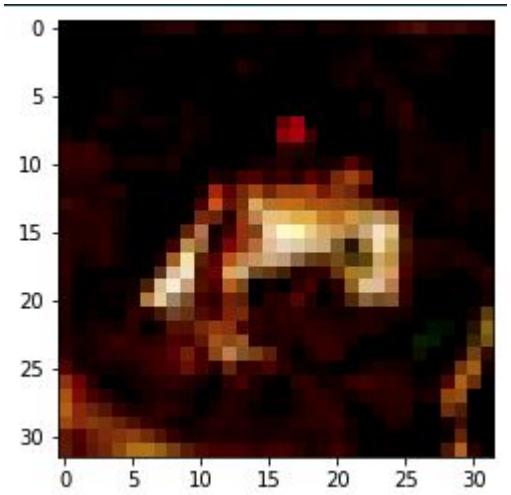
Final test accuracy = 0.9
It means 90% of test outputs correspond to labels.

The training accuracy increases with each epoch still 0.9 and the validation loss decreases with each epoch below 1.6.



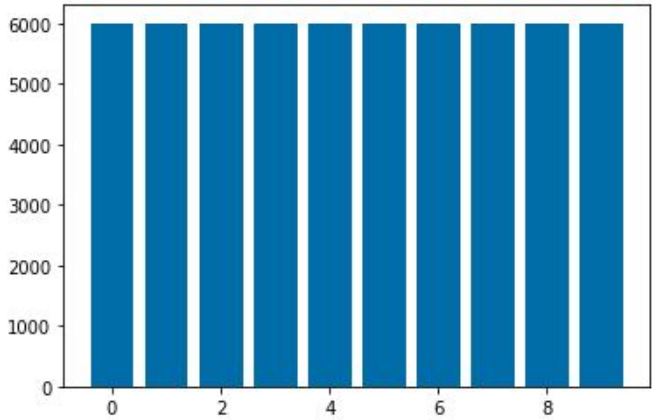


Exploratory Data Analysis CIFAR10



Training dataset composed of 50000 images of different things, and labels between 0 and 9.
Test dataset composed of 10000 images and labels.
Classes :
airplane,automobile,bird,cat,deer,dog,frog,horse,ship,truck

Shape of image : 32x32x3 colored image



perfectly
balanced dataset

CIFAR10 Classifier

The network:

	Name	Type	Params
0	layer1	Sequential	1.9 K
1	layer2	Sequential	74.1 K
2	layer3	Sequential	295 K
3	layer4	Sequential	590 K
4	layer5	Sequential	590 K
5	layer6	Sequential	590 K
6	layer7	Sequential	590 K
7	layer_linear1	Linear	2.6 K
8	sigmoid	Sigmoid	0
9	train_acc	Accuracy	0
10	valid_acc	Accuracy	0
11	test_acc	Accuracy	0

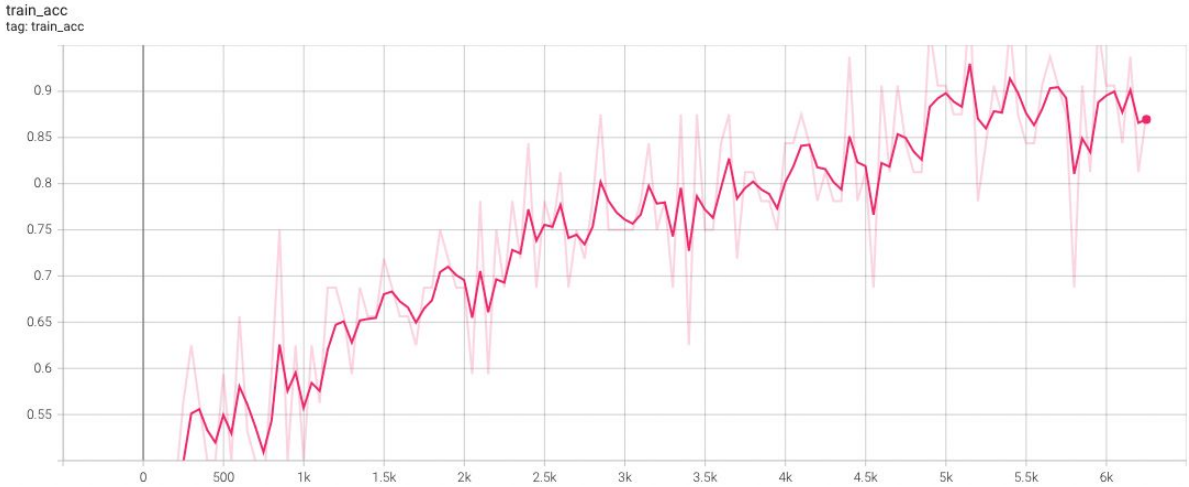
2.7 M Trainable params
0 Non-trainable params
2.7 M Total params

```
x=self.layer1(x)
x=self.layer2(x)
x=self.layer3(x)
x=self.layer4(x)
x=self.layer5(x)
x=self.layer6(x)
x=self.layer7(x)
x = x.view(x.size(0),-1)
x=self.layer_linear1(x)
```

-Input number of channels = 3 (colored image)
-Bigger network: 7 layers of Conv2d with some Maxpool2d to reduce dimensionality of the image while increasing number of channels.

Results

Final test accuracy ≈ 0.75
Can increase with the number of epoch but take too much time to do it now



With ConvNeXt

	Name	Type	Params
0	layer1	ConvNeXt	50.2 M
1	layer_linear1	Linear	10.0 K
2	train_acc	Accuracy	0
3	valid_acc	Accuracy	0
4	test_acc	Accuracy	0

50.2 M Trainable params
0 Non-trainable params
50.2 M Total params

```
x=self.layer1(x)
x=self.layer_linear1(x)
```

Final test accuracy =0.87

We just need to add a linear layer to match our vector of 10 numbers (our classes) with the output of the network.