DLCV04

Manel Baradad, Míriam Bellver, Martí Cervià, Hector Esteban, Carlos Roig

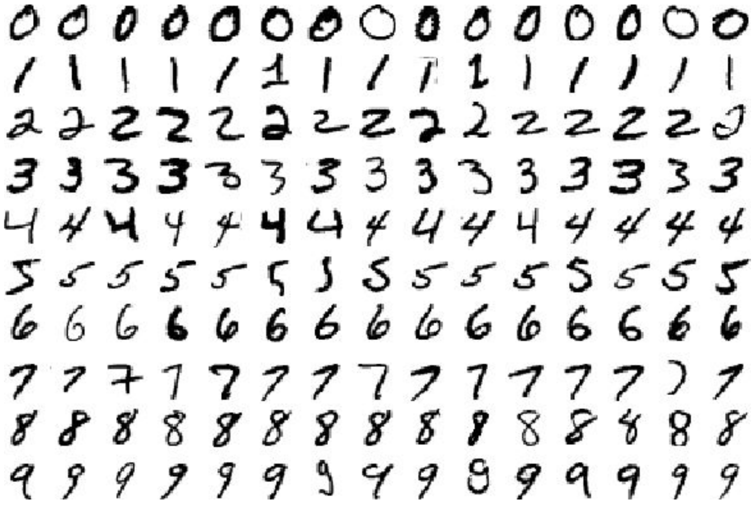
Visit our GitHub page here

Task 1: + Resources

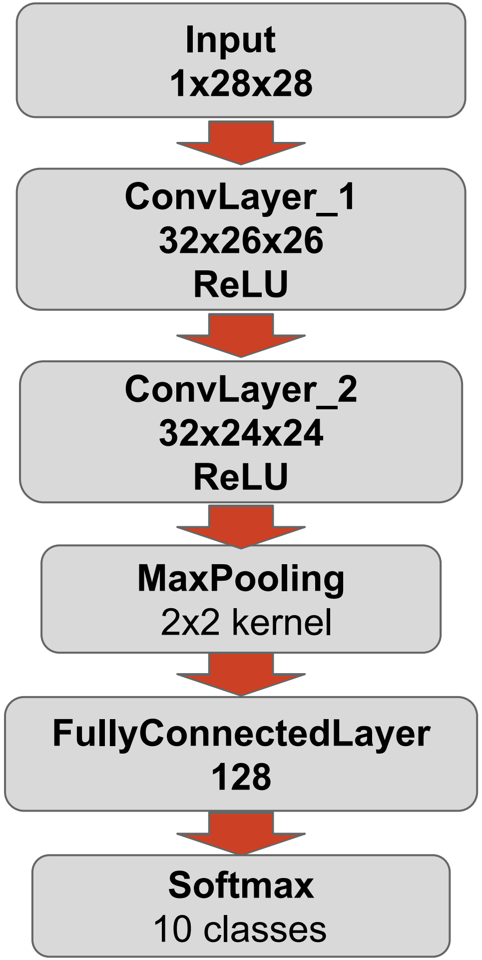
- Software used:
  - Keras with GPU accelerated Tensor Flow as backend, using python 2.7.
- Hardware used:
  - PC with GPU (NVIDIA 970)

Task 1: Architecture

For the first task, we modified a convent designed for the MNIST dataset.

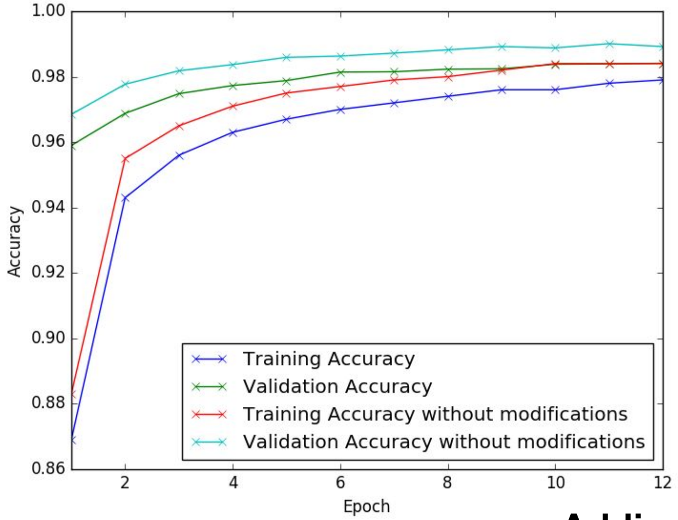


epoch time 18s parameters:600810

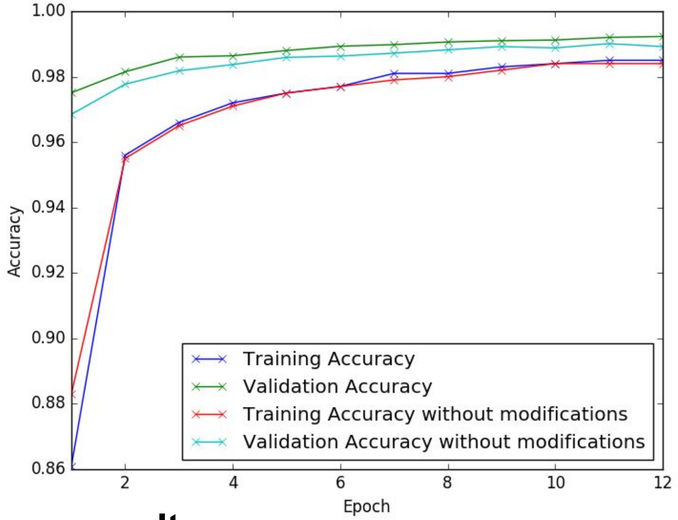


Task 1: Architecture

Removing one Conv layer



Adding 3 more Conv layers



Epoch time: 11s

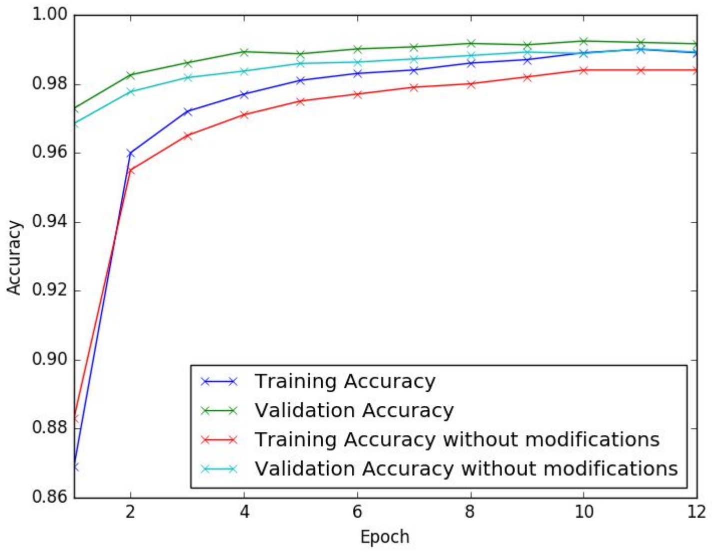
Total parameters: 693962

Adding new Conv layers results expensive in terms of total run time!

Epoch time: 39s

Total parameters: 370506

Task 1: Architecture



Epoch time: 28 sec

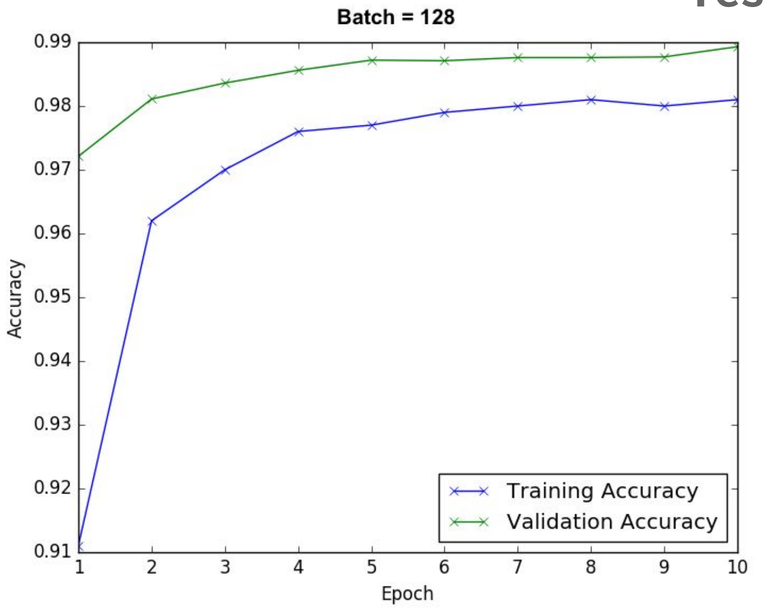
Total Parameters: 4861674

Adding a new FC layer with output size 1024 between the Conv layers and the original FC layer

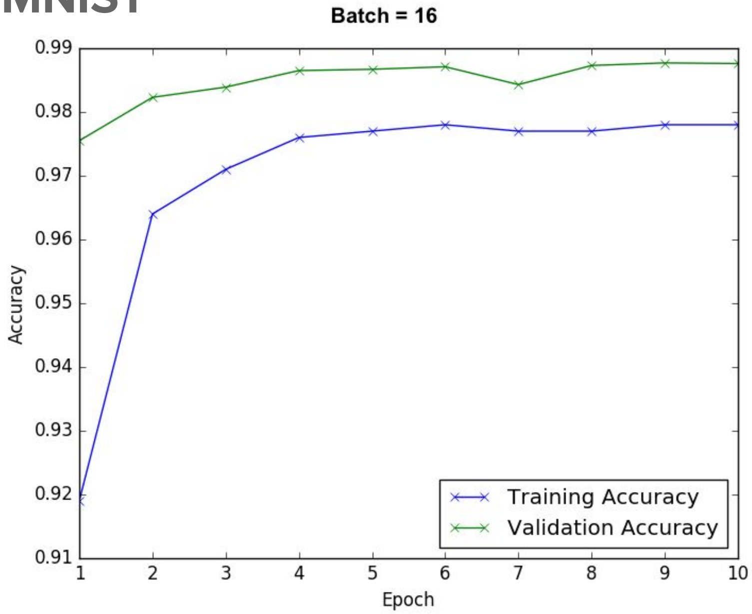
Adding new FC layers is expensive in terms of memory consumption!

Task 2: Batch Size

Test with MNIST



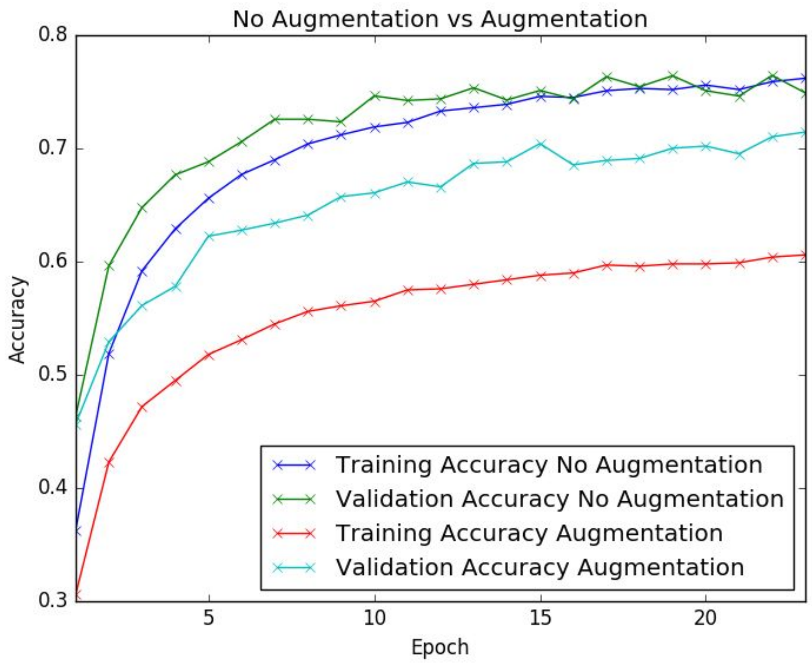
Epoch time = 28 s



Epoch time = 40 s

Only 10 classes ➔ no perceptible difference changing the batch (same happened with CIFAR10)

Task 2: Data Augmentation



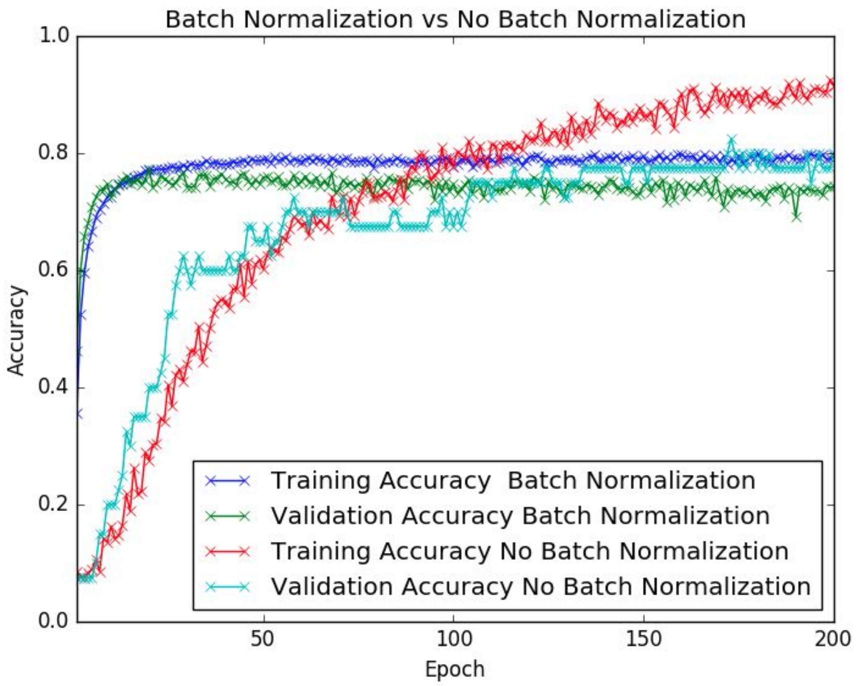
- Trained with **CIFAR10**

- Results worse with Augmentation than without Augmentation.

Possible causes:

- Not enough epochs
- This is real-time augmentation, so images are randomly flipped / translated, etc, but the amount of images per epochs is the same

Task 2: Batch Normalization



- Trained with **CIFAR10**

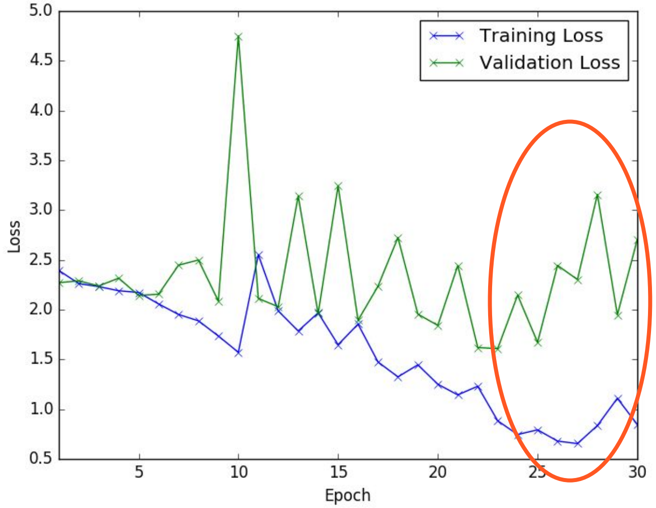
- **With Batch Normalization** the model learns **faster**, with fewer epochs

Task 2: Overfitting

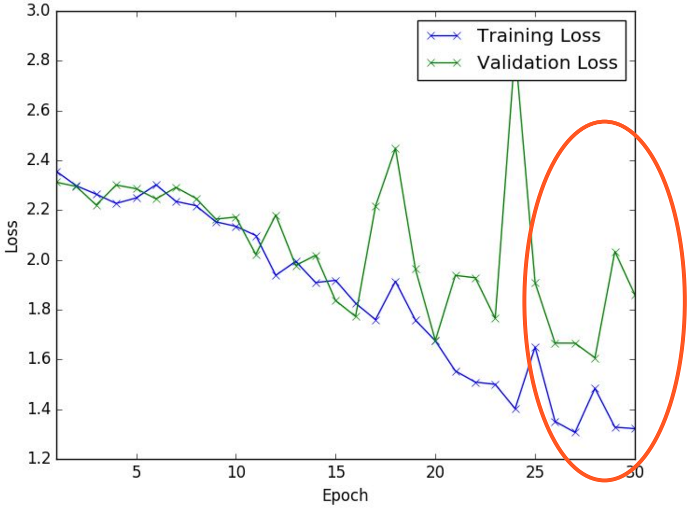
We have trained a network with 2 conv layers and 1 fully connected layer

Database: Terrassa, only 450 training images, no data augmentation

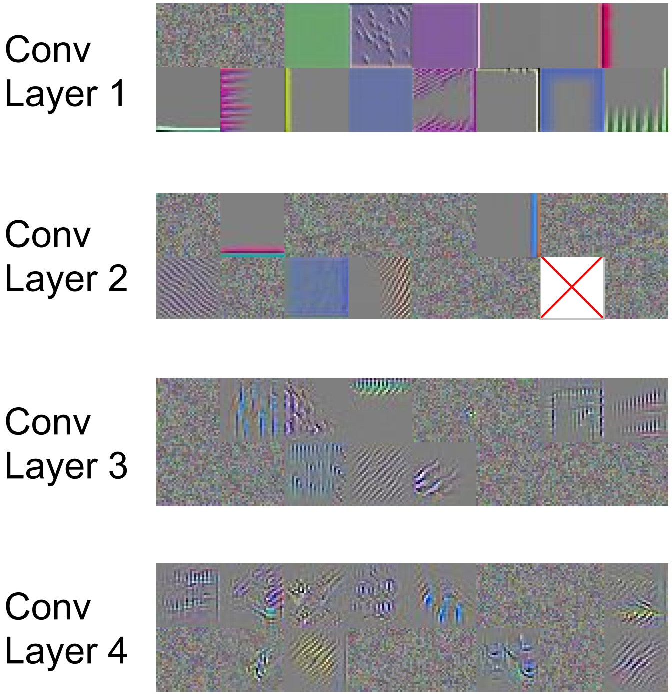
without dropout

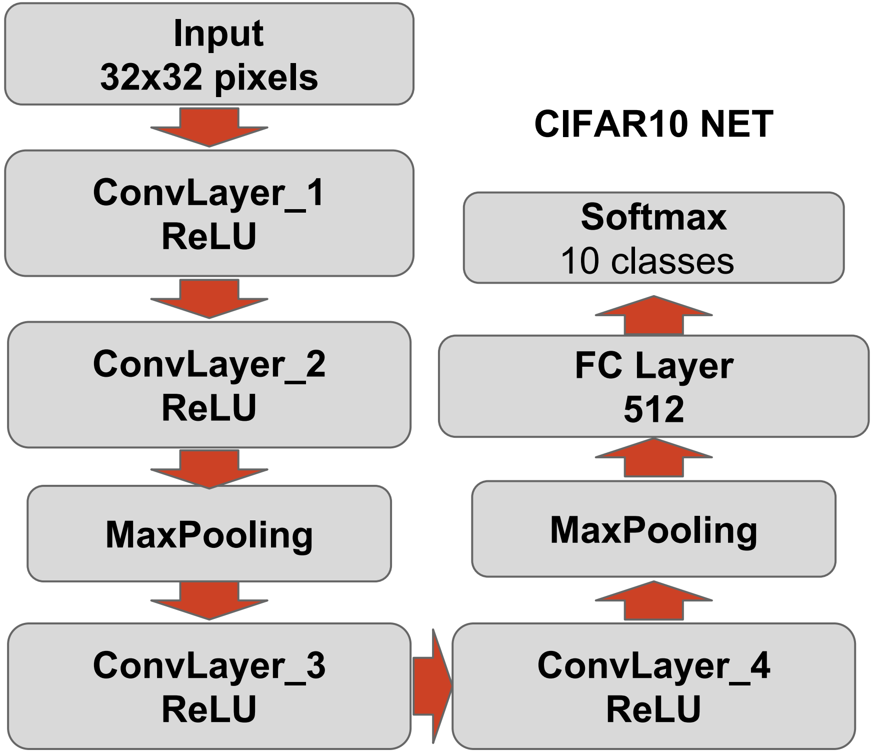


with dropout



Task 3 - Filter Visualization

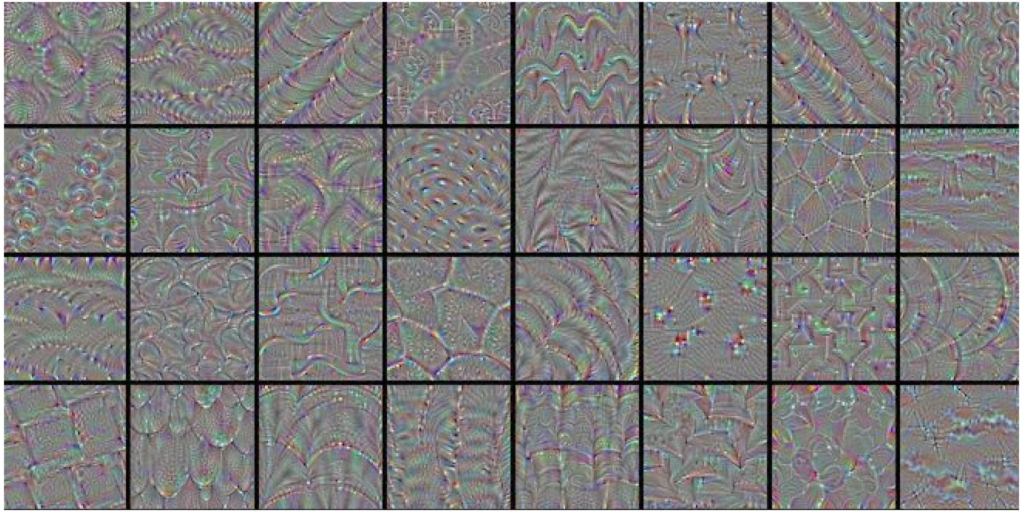




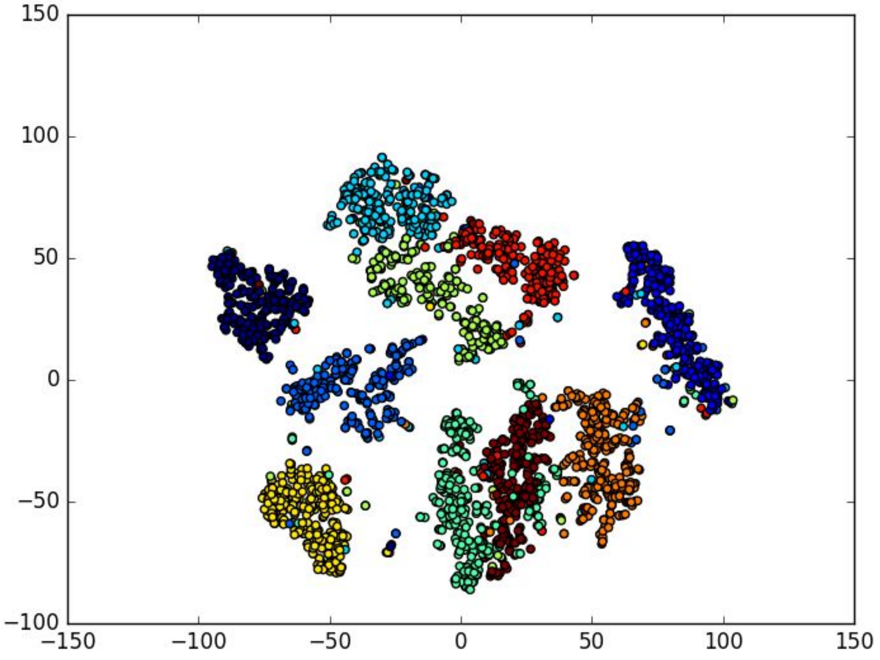
Task 3 - Filter Visualization

Filters of **pre-trained VGG16** 'conv5_1'

They are calculated defining a loss function that maximizes the activation of a specific filter in a specific layer. We have simply executed a keras/example on the following [link]



Task 3 - T-SNE



The T-SNE is a tool to visualize high-dimensional data.

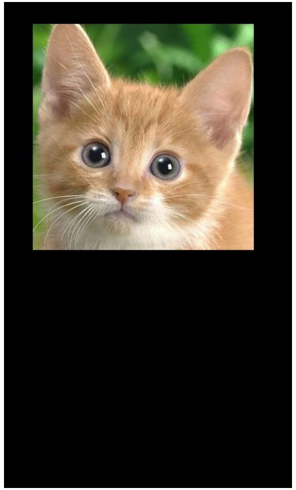
Converts similarities between data points to joint probabilities and tries to minimize the KL divergence.

In this example we used the **MNIST** dataset with 2500 images.

Task 3 - Off-the-shelf VGG-16 Local Classification

prob: 0.113

prob: 0.0493



prob: 0.25

We did these tests using a trained VGG16 network for the Imagenet dataset, the probability displayed is the value for the class of the original image.

Task 4 - Fine tuning

1. **Train network on CIFAR10 and fine-tune for Terrassa Buildings 900 2**

Terrassa database

640 labeled samples

Preprocessing

1. resize (32,32)
2. normalization color channels

CIFAR

model of Keras pre-trained 200 epochs
acc: 0.7361

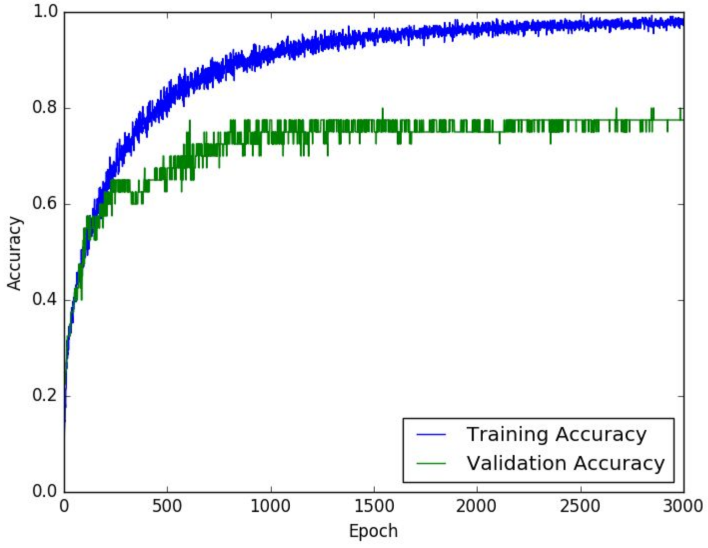
removed last layer and added a dense layer with 13 neurons

3000 epochs
batches size: 32
CIFAR fine tuning for Terrassa
val acc: 0.7750

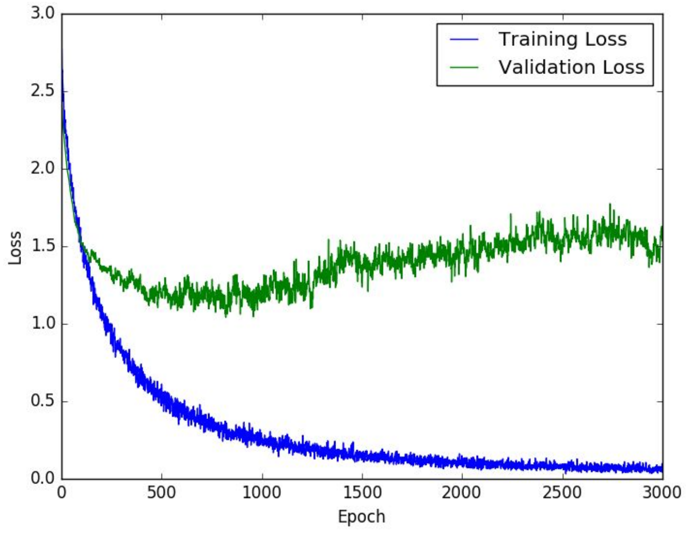
Task 4 - Fine tuning

1. Train network on CIFAR10 and fine-tune for Terrassa Buildings 900 2

Accuracy

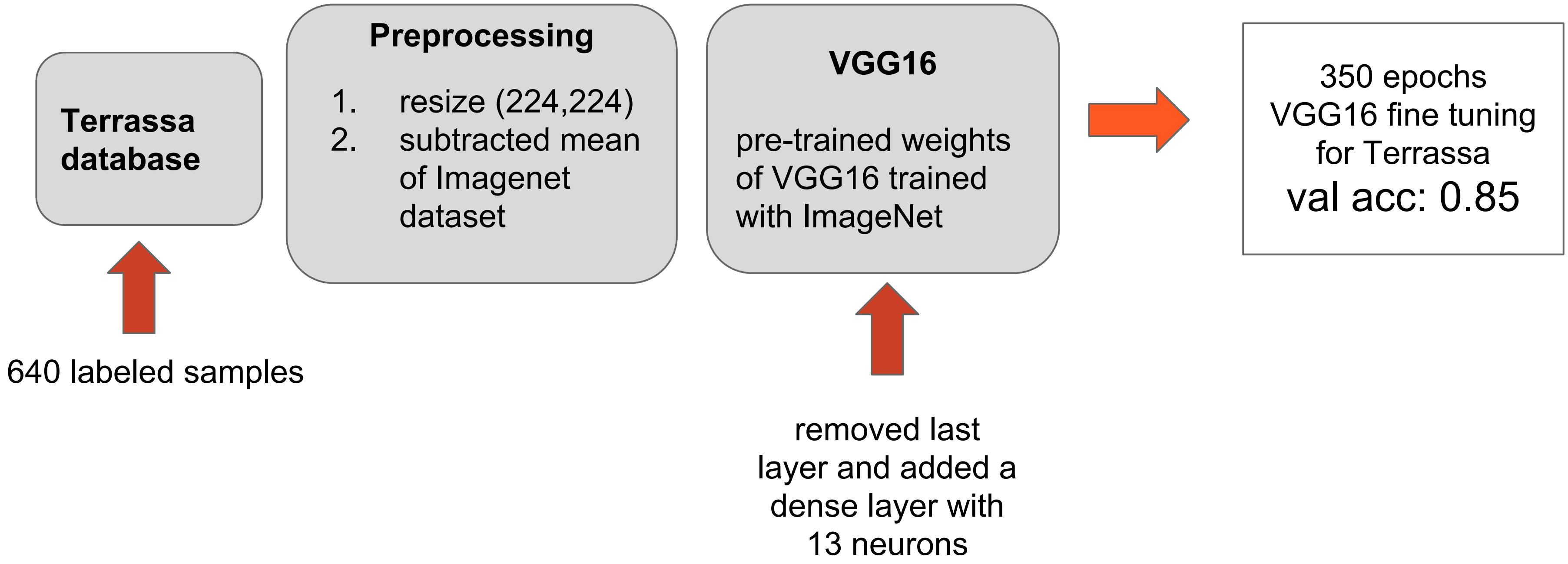


Losses



Task 4 - Fine tuning

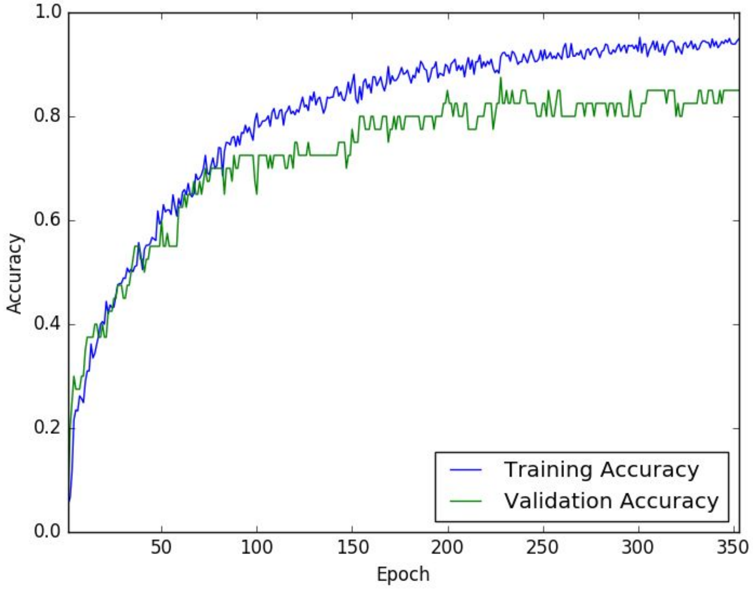
2. **Use pre-trained weights of VGG16 and train on top a classifier for Terrassa Buildings 900 2**



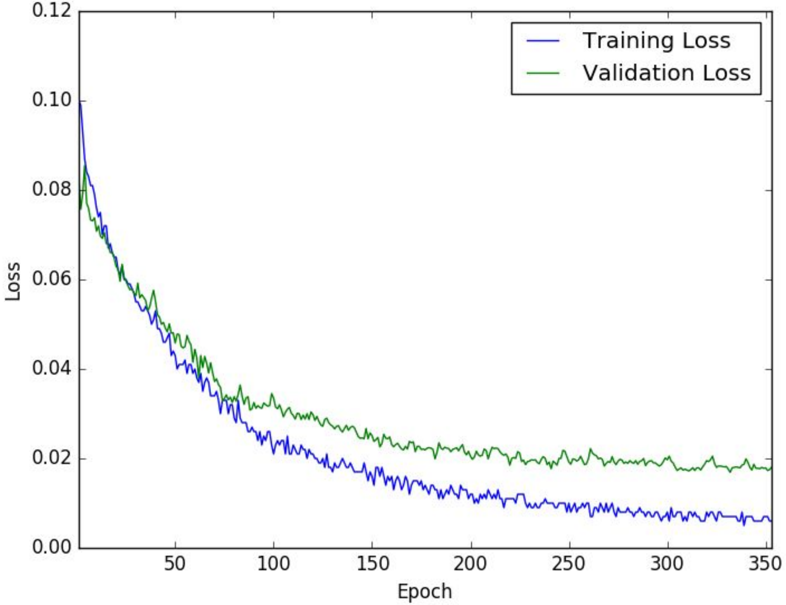
Task 4 - Fine tuning

2. Use pre-trained weights of VGG16 and train on top a classifier for Terrassa Buildings 900 2

Accuracy



Losses



Task 5 - Open Project

Neural Style

Goal: Generate a new image with the **content** of image1 and the **style** of image2

base image

style

How to encode content? ['conv4_2'] of VGG16
How to encode style? Gram matrix ['conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', 'conv5_1'] of VGG16

Task 5 - Open Project

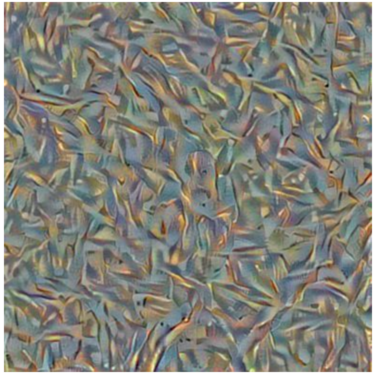
Neural Style

base image

style

default style and content features

1 iteration



10 iterations

5 iterations

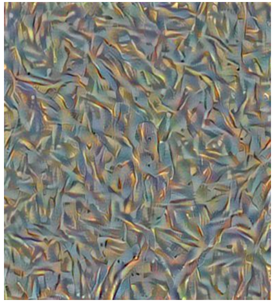
20 iterations

Task 5 - Open Project

Test 1: Only first conv style features feature_layers = ['conv1_1', 'conv2_1']

default

1



10

5

20

first conv layers features

1

10

5

20

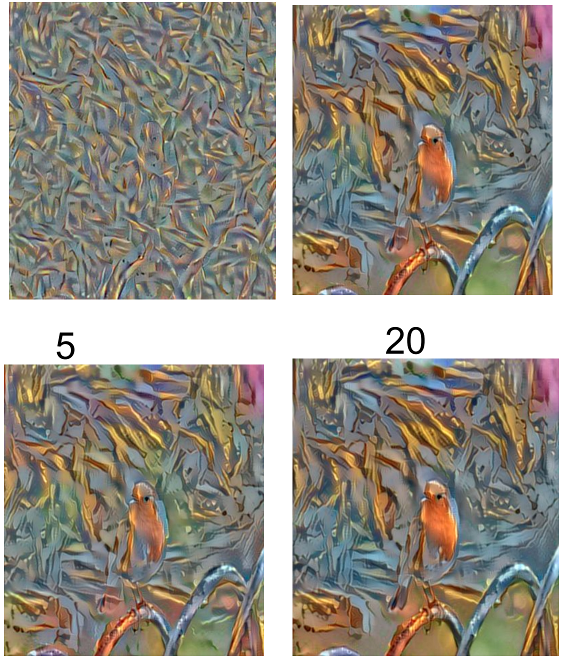
Task 5 - Open Project

Test 2: Only last conv style features feature_layers = ['conv4_1', 'conv5_1']

default

1 10

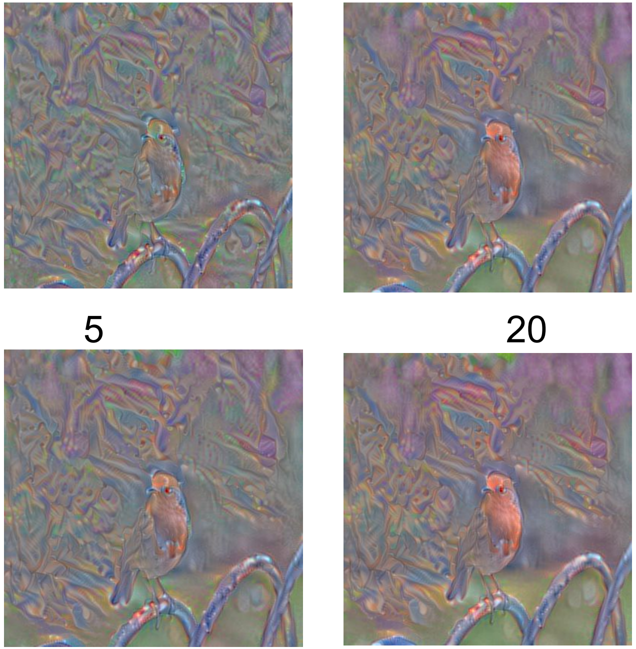
5 20



last conv layer features

1 10

5 20



Thanks!! Questions?

References

Keras

Keras repository

Keras Documentation

Visualization tool for VGG16 blog

Neural Style

A Neural Algorithm of Artistic Style

https://github.com/fchollet/keras/blob/master/examples/neural_style_transfer.py

t-SNE

Implementations on different languages