TEK5030 Deep learning lab

The **code** for this exercise can be found at
https://gitlab.com/sigmund.rolfsjord/unik4690_deep_learning_lab

To download the dataset and weights from gitlab you have to do:

```
git lfs install
```

If you cloned the repository before you installed lfs you also have to run:

```
git lfs get
```

For this exercise you need python installed. If you don't have python you can install it here:
https://docs.conda.io/en/latest/miniconda.html

You also need the python packages: **opencv-python, torch, torchvision**.
They can be installed by running:

```
pip install -r requirements.txt
```

from the base folder, or running:

```
pip install opencv-python torch torchvision
```

from anywhere. More information on how to install torch can be found here
https://pytorch.org/get-started/locally/. Here you can choose among different versions of CUDA or operating systems.

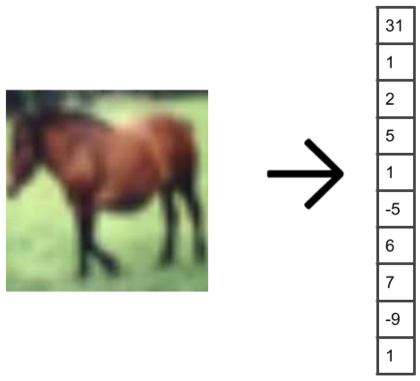
This lab is a set of exercises to learn the basics for training and running deep neural networks with python and pytorch. The lab is split in three exercises:
EX1_simple_network.py, EX2_finetune.py and EX3_live_training.py.

Go through the exercise files and solve the comments marked with TODO.

EX1_simple_network.py

In this exercise you should implement the **SimpleNet** class, to run a small network for classification of the *cifar10* dataset. The *cifar10* dataset consist of very small images from 10 different categories.

The **forward** method is called every time you run your network. This function should transform your input image, through a series of convolutions, into a vector of scores for each class.



It is important to remember that you cannot initialize your layers in the forward method, since then you will create new weight for each run and not learn anything. Use the __init__ method to initialize your layers.

Try to visualize how some the output of your layers, with cv2.imshow. Visualize both the image and the convolution result side by side, and investigate what the different layers does.

To fetch the images from torch you can do: x.data.cpu().numpy()

EX2_finetune.py

In this exercises you are still supposed to classify the *cifar10* dataset, but you should leverage that someone already trained a network to classify ImageNet. With:

self.base_model = models.resnet18(True)

you can download a pretrained network. For fast training you should only train the last layer of you network. You can either add a last layer after the output of your base network, or you can swap out self.base_model.fc, with the network you want. The model parameters are looped through and requires_grad set to *False*. This ensures that you don’t spend computational resources on calculating unnecessary gradients.

Make sure you initialize your optimizers with only the parameters you want to train.

Since the network initially are trained with 224x224 sized images, we will probably get better result by still using this size. We therefore resize our image appropriately. We also divide our image by 255, to get closer to the image type, we trained with.

EX3_live_training.py

In this exercise you should finetune a network with training images generated live from a web-camera.

We have already implemented a class named LiveDataset, where we can add images and extract training batches.

Create a training loop that adds images to the dataset, and train the last layer of the network to learn the classes we input.

Additional fun!

I also added some examples of how to run live classification and detection in: **run_detection_live.py** and **run_live.py**.