1 Computer Vision: Day 4 - Bag of Tricks to Start CNNs

Yesterday, our aim was to learn the basic principles of constructing convolutional neural networks. This was just enough to solve the task at hand. Today, we will expand the networks and tune their performance and training. You will use ways to enrich your training data and make the nets more robust, reduce the training time and convergence rate, and find out what the neural model focuses on for performing recognition.

1.1 Libraries

For today, the following libraries can and should be used to work on the exercises:

- `six.moves.cPickle` — a library for fast serialization
- `numpy` — a library for efficient matrix operations
- `matplotlib.pyplot` — a library for displaying images and plotting graphs
- `cv2` — a library for basic image processing operations
- `scipy` — a library for scientific computations
- `sklearn` — a library for machine learning
- `joblib` — a library for memoisation
- `tensorflow` — a library for deep learning

You can use other libraries that are installed on the lab computers.

1.2 Exercises

You can and are encouraged to reuse code and maybe trained models from the days before.

1. Define and train or reuse a multi-layer network with convolutional layers, max-pooling layers and in the end fully connected layers and evaluate its performance.
2. Initialize the weights with Xavier initialization.
3. Think of some useful data augmentation and apply it.
4. Tune the batch size and learning rate of your training.
5. Use dropout.
6. Use some early stopping strategy.

1.3 Hints

1. Until you are sure that your graph is training correctly, it is advised to only train on a small toy example.
2. `tensorflow` works a lot like `numpy`, just for symbolic variables. Many functions even carry the same name.

1.4 Functions

Some of the following functions may or may not come in handy when working on the exercises. Have a look at the documentation if they seem worth the while.

`tensorflow.contrib.layers.batch_norm` — adds a batch normalization layer
`tensorflow.nn.dropout` — adds a dropout layer