Introduction to Deep Learning

Chapter 4: CNN: Application

Bernd Bischl

Department of Statistics – LMU Munich

WS 2021/2022

APPLICATION - IMAGE CLASSIFICATION

- One of use case for CNNs is image classification.
- There exist a broad variety of battle-proven image classification architecture such as the AlexNet , the Inception Net or the ResNet which will be discussed in detail in the next lecture.
- All those architectures rely on a set of subsequent convolutional filters and aim to learn the mapping from an image to a probability score over a set of classes.

APPLICATION - IMAGE CLASSIFICATION

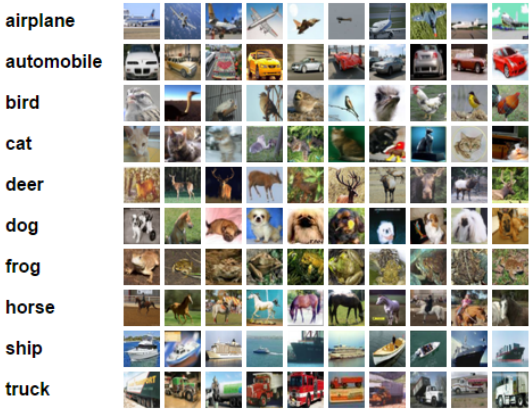


Figure: Image classification with Cifar 10: famous benchmark dataset with 60000 images and 10 classes (Alex Krizhevsky (2009)). There is also a much more difficult version with 60000 images and 100 classes.

APPLICATION - IMAGE CLASSIFICATION

Figure: One example of the Cifar10 data: A highly pixelated, coloured image of a frog with dimension [32, 32, 3].

APPLICATION - IMAGE CLASSIFICATION

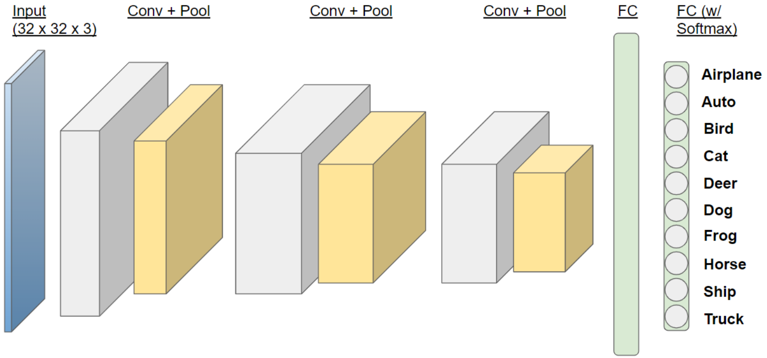


Figure: An example of a CNN architecture for classification on the Cifar10 dataset (FC = Fully Connected).

CNN VS A FULLY CONNECTED NET ON CIFAR10

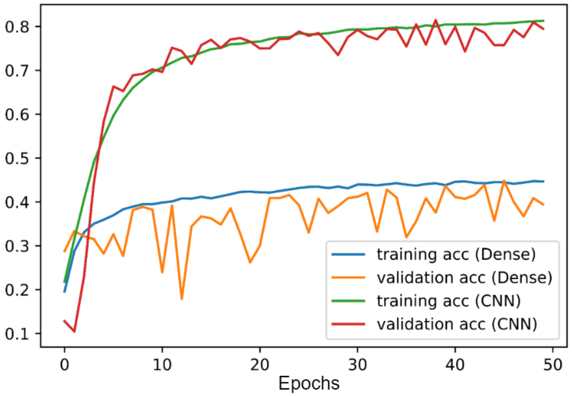


Figure: Performance of a CNN and a fully connected neural net ("Dense") on Cifar-10. Both networks have roughly the same number of layers. They were trained using the same learning rate, weight decay and dropout rate. Of course, the CNN performs better because it has less learnable parameters and the right inductive bias for the task.

APPLICATION - IMAGE COLORIZATION

- Basic idea (introduced by Zhang et al., 2016):
  - Train the net on pairs of grayscale and coloured images.
  - Force it to make a prediction on the colour-value **for each pixel** in the grayscale input image.
  - Combine the grayscale-input with the colour-output to yield a colorized image.
- Very comprehensive material on the method is provided on the author's website. ▸ Click here

APPLICATION - IMAGE COLORIZATION

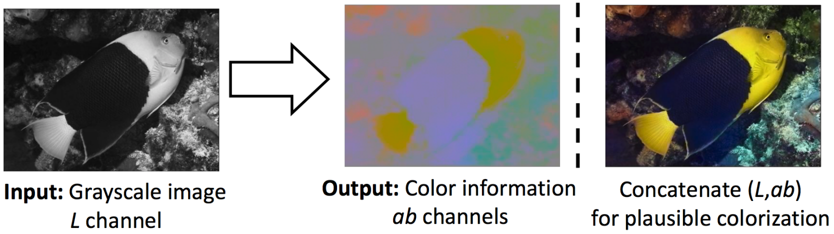

Input: Grayscale image
L channel
Output: Color information
ab channels
Concatenate (*L*,*ab*)
for plausible colorization

Figure: The CNN learns the mapping from grayscale (L) to color (ab) for each pixel in the image. The L and ab maps are then concatenated to yield the colorized image. The authors use the LAB color space for the image representation.

APPLICATION - IMAGE COLORIZATION

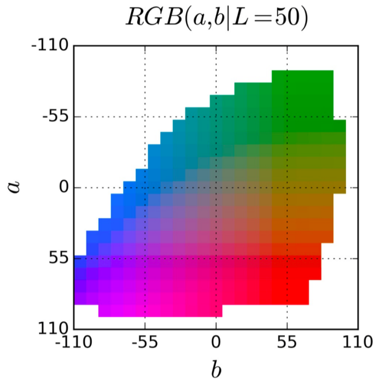


Figure: The colour space (ab) is quantized in a total of 313 bins. This allows to treat the color prediction as a classification problem where each pixel is assigned a probability distribution over the 313 bins and that with the highest softmax-score is taken as predicted color value. The bin is then mapped back to the corresponding, numeric (a,b) values. The network is optimized using a multinomial cross entropy loss over the 313 quantized (a,b) bins.

APPLICATION - IMAGE COLORIZATION

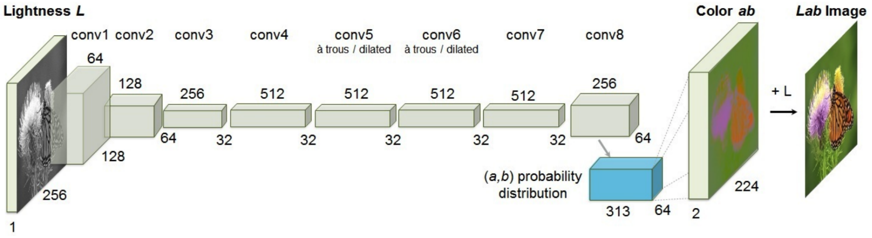


Figure: The architecture consists of stacked CNN layers which are upsampled towards the end of the net. It makes use of *dilated convolutions* and *upsampling layers* which are explained in the next lecture. The output is a tensor of dimension [64, 64, 313] that stores the 313 probabilities for each element of the final, downsampled 64x64 feature maps.

APPLICATION - IMAGE COLORIZATION

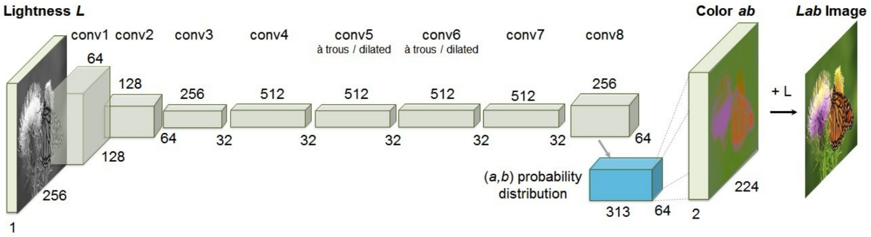


Figure: This block is then upsampled to a dimension of 224x224 and the predicted color bins are mapped back to the (a,b) values yielding a depth of 2. Finally, the L and the ab maps are concatenated to yield a colored image.

APPLICATION - OBJECT LOCALIZATION

- Until now, we used CNNs for *single*-class classification of images - **which object is on the image?**
- Now we extend this framework - **is there an object in the image and if yes, where and which?**

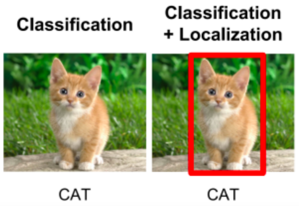


Figure: Classify and detect the location of the cat.

APPLICATION - OBJECT LOCALIZATION

- Bounding boxes can be defined by the location of the left lower corner as well as the height and width of the box: $[b_x, b_y, b_h, b_w]$.
- We now combine three tasks (detection, classification and localization) in one architecture.
- This can be done by adjusting the label output of the net.
- Imagine a task with three classes (cat, car, frog).
- In standard classification we would have:

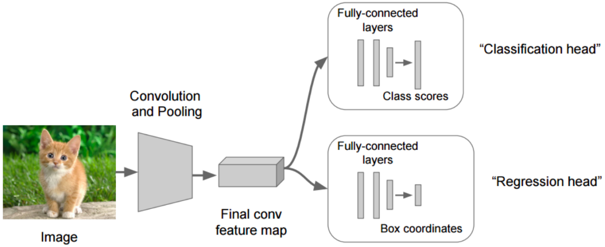
$$\text{label vector} \begin{bmatrix} c_{cat} \\ c_{car} \\ c_{frog} \end{bmatrix} \text{ and softmax output } \begin{bmatrix} P(y = cat|\ X, \theta) \\ P(y = car|\ X, \theta) \\ P(y = frog|\ X, \theta) \end{bmatrix}$$

APPLICATION - OBJECT LOCALIZATION

- We include the information, if there is a object as well as the bounding box parametrization in the label vector.
- This gives us the following label vector:

$$\begin{bmatrix} b_x \\ b_y \\ b_h \\ b_w . \\ c_o \\ c_{cat} \\ c_{car} \\ c_{frog} \end{bmatrix} = \begin{bmatrix} \text{x coordinate box} \\ \text{y coordinate box} \\ \text{height box} \\ \text{width box.} \\ \text{presence of object, binary} \\ \text{class cat, one-hot} \\ \text{class car, one-hot} \\ \text{class frog, one-hot} \end{bmatrix}$$

APPLICATION - OBJECT LOCALIZATION



- Naive approach: use a CNN with two heads, one for the class classification and one for the bounding box regression.
- But: What happens, if there are two cats in the image?
- Different approaches: "Region-based" CNNs (R-CNN, Fast R-CNN and Faster R-CNN) and "single-shot" CNNs (SSD and YOLO).