Usable ML Writeup

Convolutional Layer:

As the key components in a CNN, the convolutional layers perform two jobs: extract key features from the input while reducing its dimensionality, i.e. its size. To do this, a matrix smaller than the input is randomly initialized and iterates over the pixels while performing matrix multiplication with the given pixels. Depending on the matrix this might result in the detection of certain edges or shapes. The output is a so-called feature map with highlighted features. In each convolutional layer there are a multitude of these filters, each adjusted individually during training to improve performance.

Parameters:

Kernel size determines the size of the filter / kernel. A larger kernel reduces the dimensionality further, at the same time more information is lost.

Stride determines the distance the kernel travels over input. A stride of 1 means it passes over each pixel in the matrix, a stride of 2 every other pixel and so on.

Padding adds additional fields around the edges. This way, the area on the edges is larger and more data can be extracted from it.

Maxpool layer:
Maxpooling is another operation that reduces dimensionality, thereby decreasing the number of calculations necessary. It is best explained by an example: Consider a 4x4 matrix. This matrix can be split into four quadrants of size 2x2. Maxpooling chooses the largest value in each quadrant (hence max pooling) and returns a 2x2 matrix with these values.

Linear layer:
Linear layers make up the final layers in neural networks for classification. As input, they take a vector which in a CNN is a flattened matrix from the previous layer. While data passes through the network, the linear layers perform a linear transformation by multiplying each input neuron by a weight and adding a bias. Several of these layers can be connected sequentially to reduce the number of inputs. Finally, the output consists of a neuron for each class and a corresponding probability determining how likely the input belongs to each class.

Activation functions:

Activations functions play a critical in neural networks as they add nonlinearity to the models hidden layers. This is necessary as it allows for more complex functions inside the model that can also capture nonlinear relationships. Typically, the same activation function is chosen for the entire network.

ReLU:
Arguably the simplest and one of the most used activation functions is the rectified linear unit or ReLU. It returns the input if the input is positive and a zero if it's negative.

Tanh:
Hyperbolic tangent or simply Tanh squashes an input between -1 and 1 where very negative values tend towards -1 and vice versa. Works well with inputs that have already been scaled to be between-1 to 1.

Sigmoid
The sigmoid or logistic function is similar in nature to the Tanh function, the only difference being that values are squashed between 0 and 1. Also works well with data scaled to its respective range.

Training hyperparameters

Epochs:

How often the learning algorithm works through the entire training dataset.

Learning rate:

By how much to adjust the internal parameters of a neural network during training. Choosing a learning rate that's too large will lead to over- or undershooting the true optimum while a learning rate that's too small will increase the time to see improvements dramatically.

Batch size:

How many samples to run through before updating the model parameters. Done both to preserve memory (large datasets can be split into several batches) and improve optimization (individual samples can be noisy, regularizes over a batch of samples).