## Why do general machine learning algorithms (vector input) perform poorly on images?

General machine learning algorithms with vector input do not account for the spatial relations and arbitrariness (of object position, angle, and lighting condition) in image data. Spatially close pixels are highly correlated, e.g., the belong to the same object with a high probability and have same intensity/ colour with a high probability. Furthermore, object can appear anywhere in an image from any angle and lighting condition. A good feature extractor should incorporate these properties of image data. A machine learning algorithm that does so is a convolutional neural network (CNN).

CNNs do so by utilizing tensor inputs, pattern matching, and channels. The input of a CNN is a tensor of shape (3 channels for RGB or 1 for greyscale). Furthermore, CNNs use local connectivity, meaning a neuron in a CNN having only a small receptive field. This way, the spatial relations of the input are considered. Larger or global receptive fields are eventually achieved by stacking layers enlarging receptive fields with each layer in a pyramid like manner. Furthermore, in this hierarchical manner low-level features are used to aggregate high-level features (for explanation see below).

Each neuron in the same channel looks for the same pattern, this is achieved by weight sharing. This way an object can be spotted anywhere in an image. To get different patterns yielding different information in one layer different channels are used (those can be imagined as different family of neurons, in one family all neurons share the same weight, but each family of neurons see the whole output of the previous layer, even though a single neuron only sees it receptive field, determined by its local connectivity) As a consequence, the problem of arbitrariness of object position, angle, and lighting condition in image data is tackled.

## Explain the terms feature, low-level feature, and high-level feature.

A feature is an abstract representation of a property yielding information which is directly or indirectly relevant for a given task. Here, indirectly means that features on a lower level may constitute higher level features in a hierarchical manner with the highest level yielding the information for the given task. Neural networks learn features at each layer. Lower layers learn less abstract features while higher layers learn more abstract features. For example, the first layer detects edges, combination of edges form motifs, motifs form parts in higher layers, and parts form objects in the last layers. These objects might than be the classes used by a linear classifier in a classification task.

## What is the purpose of feature extraction?

Feature extraction in general is the task of obtaining features from input data yielding information which is directly or indirectly relevant for a given task. We can distinguish between feature extraction of hand-crafted features and features obtained by representation learning. Feature extraction of hand-crafted features is rule-based. This is done in classical machine learning. However, these hand-crafted features do not yield significant information for all tasks, especially for complex tasks these hand-crafted features are lacking. For example, what features would be sufficient to distinguish a songbird from an eagle? The beak, the size, the colour of features? Furthermore, these features are high level, but hand-crafted features are usually low-level, like edge detection. Accordingly, we do not know how to compute these high-level features from pixel data? In summary, we can not hand-craft reliable high-level features. Therefore, we try to learn them. This approach of learning features is called representation learning

## What is the purpose of a loss function?

In machine learning the parameters of a model are not set by hand but are learned. The loss function is used to train a model. The model is trained by optimizing the loss function with regard to the parameters of a model. Thus, a loss function needs to measure performance of the model on a dataset . The convention is that lower loss means better performance. Performance should be a measurement which meaningfully describes how much the goal of a given task is achieved by the model. A common loss function for classification is the cross-entropy loss.

## What does the cross-entropy measure?

Given two mass function , The cross-entropy measures the dissimilarity of those two mass functions. The more dissimilar , the higher . Note that can only become 0 if .

## Which criteria must the ground-truth labels and predicted class-scores fulfill to support the cross-entropy loss, and how is this ensured?

As stated in the definition the ground-truth labels and prediction class-scores must be mass functions to support the cross-entropy loss. A mass function is a function that complies to the following rules: and .

For the ground-truth labels this is ensured by one-hot-encoding. For the predicted class-scores this is ensured by the softmax function. A one-hot encoded label is a vector of size , with being the number of classes. Each element (index) of that vector corresponds to one class. In a one-hot encoded vector only the element at the index of the class is all other elements are . The predicted class-score follows the same index mapping and the softmax function takes an arbitrary vector and turns it into a mass function. The softmax function is defined by . In general, and something greater equal 0 divided by the sum of things greater equal 0 is also greater equal 0 (to prevent division by 0, usually a small delta is added to the sum). Hence, the softmax function complies with . Furthermore, each element of the softmax function is normalized by . In consequence, . Accordingly, the output of a softmax function is a mass function. For these reasons, ground-truth labels and predicted class-scores fulfill to support the cross-entropy loss.

## What is the purpose of the training, validation, and test sets and why do we need all of them?

The training dataset is used to train a neural network. The validation dataset and the test dataset are not used for training. On that account, the neural network cannot overfit to the validation dataset and the test dataset. Thus, the validation dataset can be used to explore different hyperparameters and monitor whether or not the neural network overfits to the training dataset. Based on the performance on the validation dataset, the neural network is chosen for evaluation. This way, the neural network might develop a dependency on the validation dataset. For example, a neural network might perform well on the validation dataset by chance but less well on other data. For this reason, the neural network is evaluated only once on the test dataset. The performance of the neural network on the test dataset is regarded as the performance of the neural network.

## Results

Plot of validation accuracies over epochs

Chart, line chart

Description automatically generated

Best validation accuracy:

Test accuracy:

As discussed, the validation dataset and the test dataset are not used for training. The validation dataset is used for hyperparameter exploration. This way, the neural network might develop a dependency on the validation dataset. For this reason, a test set is used to get an unbiased evaluation of the final modal. Consequently, usually, we would expect the validation accuracy to be higher than the test accuracy, However, the only “hyperparameter” we tuned was the number of epochs, i.e., for evaluation we chose the model from the epoch with the best validation accuracy.

Accordingly, it is likely, that the model did not develop a big dependency on the validation data. Since both sets are chosen random from the same data distribution, they should follow a similar distribution. Thus, it is to be expected that the validation and test accuracy are similar (no statistically significant difference). This is shown in our results, best validation accuracy and test accuracy are exactly equal, i.e., 0.6065.

## Optimizer and Params

We used the Adam optimizer with default pytorch parameters *learning rate=0.001*, *betas=(0.9, 0.999)*, *epsilon=1e-08*, *weight\_decay=0*, *amsgrad=False*, *\**, *maximize=False*.