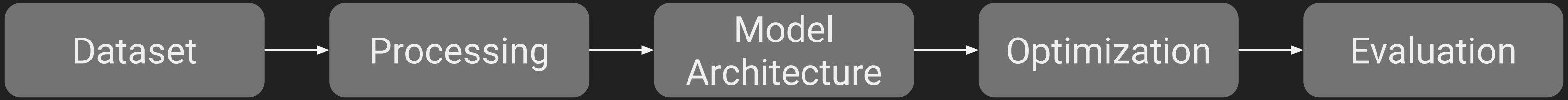
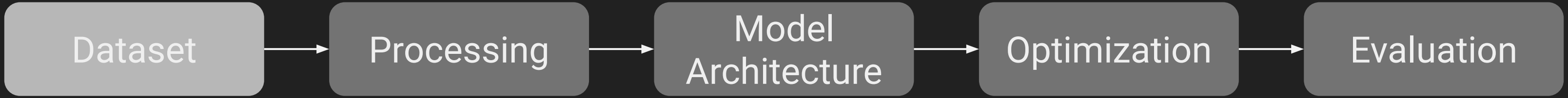
Machine Learning Pipeline

Intro

There are several components to a machine learning code and it is helpful to talk about the organization of the code before diving into the specifics of libraries like Tensorflow. The code can also become very messy, and we will talk about how to split up the program for best results and sanity.

Overview


Dataset
Processing
Model Architecture
Optimization
Evaluation

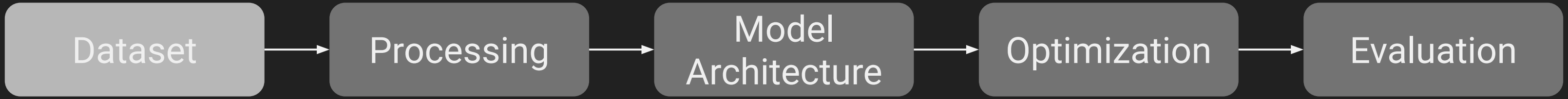


You would most likely use supervised learning in this class, which means you will have labeled data.

The dataset can come in various formats:

- csv, .tsv
- tfrecords
- .mat arrays
- .pngs

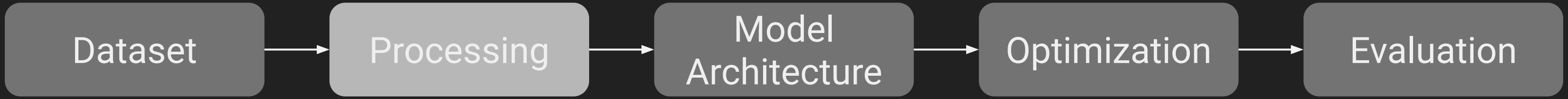
and more..



You need to read in your dataset as the first step, and then have a method to ‘consume’ the data.

Luckily, (or maybe not) Tensorflow has the `tf.data` API which allows you to read and consume data from various sources

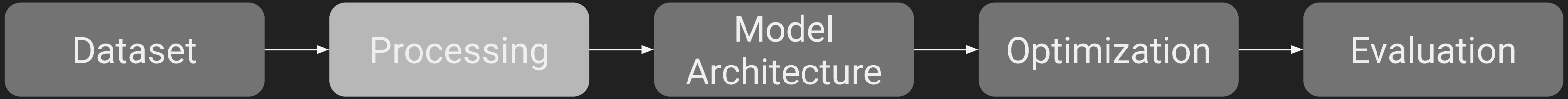
Sometimes, with very large datasets, data needs to be loaded in chunks



Many times, you would want to have some pre-processing on your data before you give it to the model

This may include:

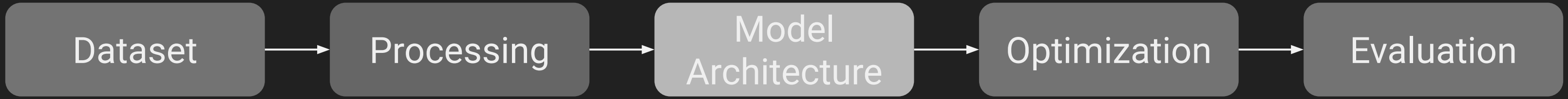
- Feature Extraction
- Handling missing data
- Normalization
- Data Augmentation
- Scaling and/or resizing images



You need to decide when to do this processing if it is needed

Runtime Processing vs **Batch Processing**: Trading storage against compute cost

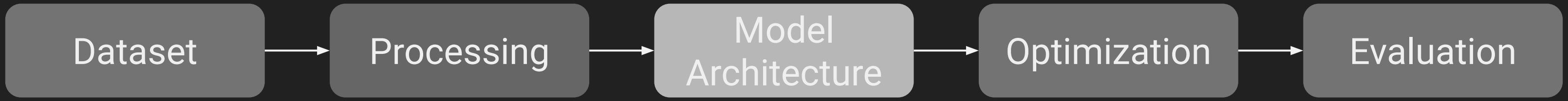
You can use `tf.data` API to apply pre-processing functions to your data



This is obviously a very important piece. You can either choose a standard architecture like:

- AlexNet CNN
- Residual CNN
- DenseNet CNN

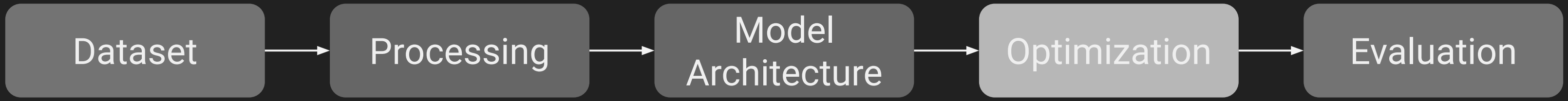
Or you can make your own.



There are several parameters you can choose in these networks:

- Network depth
- Filter size
- Activation Function

You will use a lot of `tf.keras.layers` and most likely also the `tf.keras.Model` API

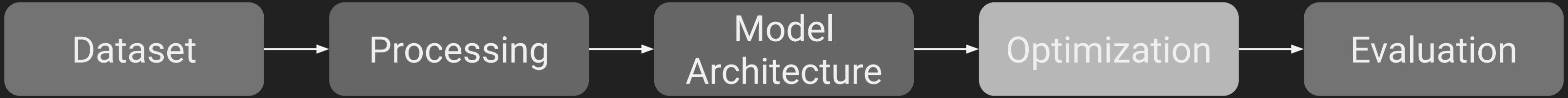


Optimization consists of two things: choosing a Loss Function and choosing an Optimizer.

We choose a loss function depending on the task we are performing. We also have a choice of using regularizers. Some common loss functions are:

- Cross Entropy Loss (usually for classification and segmentation)
- Mean Squared Error (reconstructions)
- Mean Absolute Error (reconstructions)

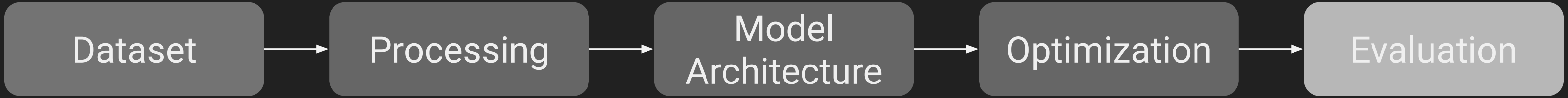
These are available in `tf.keras.losses`



Optimizers are flavours of Gradient Descent, the ones available are in `tf.optimizers`. Some examples are:

- Adam (most popular)
- SGD
- RMSprop

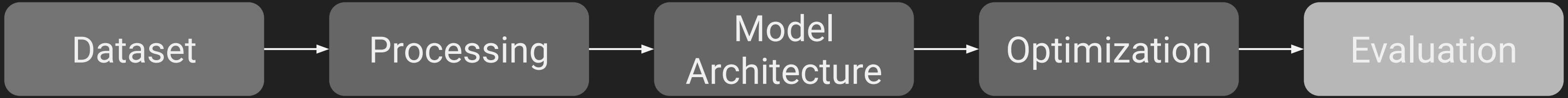
For optimizers, the most important hyperparameter is the learning rate. Tuning the learning rate affects the performance in a big way. We can have **learning rate schedules** too.



Once your optimization is done, you would want to evaluate the performance. There are various performance indicators that you might consider:

- Value of the loss
- Accuracy
- Sensitivity, Specificity
- Precision, Recall
- IOU

These are available in `tf.metrics`



Finally, you should test your model on unseen data.

To do this, you have change your input source to use the test data

You have to change some model parameters to test mode (like batch norm or dropout)

You will not use optimization, instead you will take the model output and apply the metrics of your choice on them