## LHL-Final-Project: Rabbit Breed Classifier- Flask Web App

# Motivation:

I wanted to pick a project which could help me assess if I have learned the art of learning. I chose something that brought together:

* My love for bunnies and their adorable ways.
* A focus on creating a visually engaging app.
* A dive into deep learning to really test and grow my tech skills.

Task: Image classification of 4 Rabbit Breeds using convolutional Neural Network

**Tools Used**

* **Notebook**: Google Colab for model development and training.
* **Web Framework**: Flask, with the development environment set up in Visual Studio Code.
* **Data Collection**: Bulk image download in a ZIP file format.

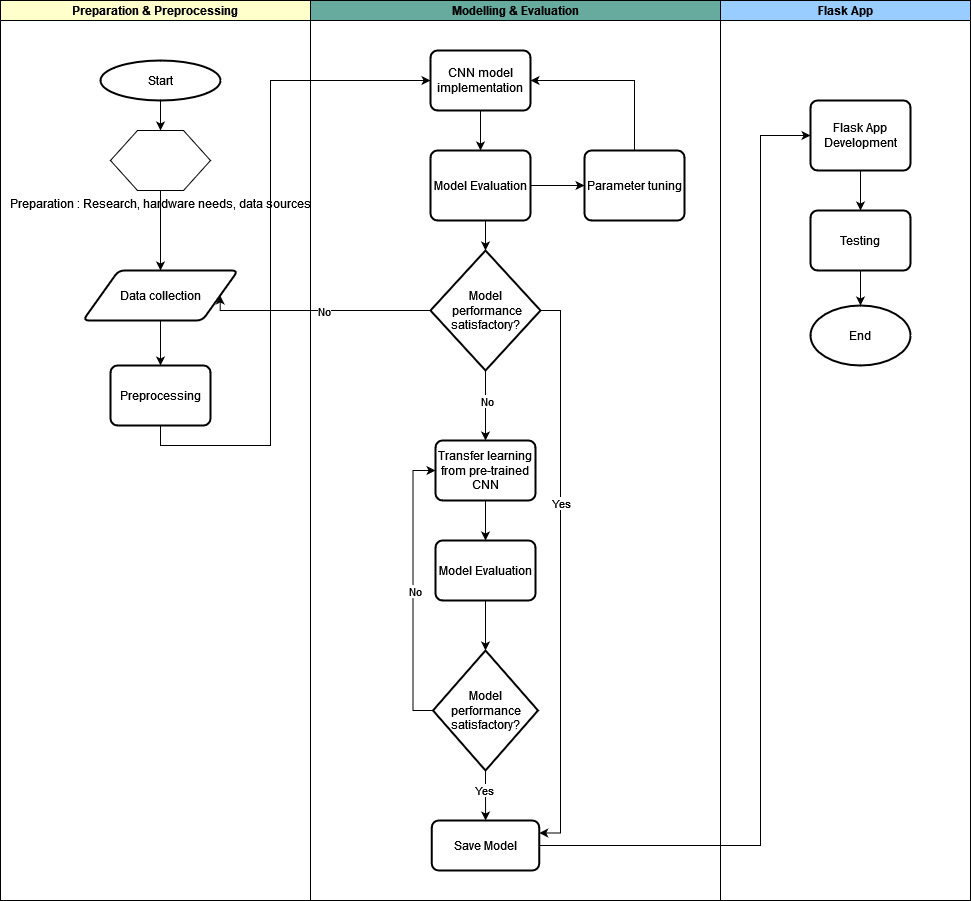
# Repository Guide

The project comprises several key components designed to work in harmony:

1. **Rabbit Breed Dataset**: This includes comprehensive data on all the rabbit breeds I have interacted with, forming the foundation for the model training.
2. **Modeling Notebook**: Utilizing Google Colab, I've developed and refined my deep learning model in an accessible and powerful cloud-based environment.
3. **Image Classification Application**:
   * **Static Folder**: Contains the static assets required for the web application.
   * **Templates**: Features the index.html file which serves as the front-end interface for user interaction.
   * **Trained Model File**: rabbit\_breeds\_classifier\_vgg.h5, the heart of the app, which is the trained VGG16 deep learning model.
   * **Flask Application Script**: flask\_app.py is the backbone script that uses Flask to serve the model and handle the web app's functionality.

# Process Outline

Shared below is the process workflow:



### # Preparation

* Research on CNN architectures.
* Decision on computational resources - chose Google Colab for cloud GPU access
* Vetting data that can be used for the project

# Data collection

Sources: Google Images, Facebook Rabbit Community, Instagram  
Number of images per breed: <br>

California: 104  
Dutch: 108  
Holland Lop: 103  
Lionhead: 105

## # Pre-processing:

## Pre-processing involved image resizing, Image resizing, normalization, label encoding (one-hot), shuffling and adding weights (due to imbalance in classes).

## # CNN Model trained from scratch

## I began with training a CNN model from scratch to provide a starting point and a baseline for my project. I kept the model simple to avoid over-fitting. An illustration of the model is shared below (for details on the structure, please refer to the notebook).

## I used a small batch-size so that the model can learn from each individual example although it takes longer to train. A larger batch size trains faster but it may result in the model not capturing the nuances in the data.

## The model performance was then evaluated against test and validation data and through the metrics of accuracy and loss. The model performance was unsatisfactory (refer to the results section) and so then I proceeded to incorporate transfer learning through a pre-trained model.

## # Transfer learning model

## For this model, I added in custom layers and used ‘freeze the weights’ technique to to avoid destroying any of the information contained by base layers during the first few epochs of training. I then evaluated this model using the same metrics as the first model.

## # Results

## CNN- Model Trained from Scratch

The Model is giving us a test accuracy of 50% while the loss is 1.12. Ideally want the loss to be as close to 0 as possible and the accuracy to be as high as possible. We will now look at plots of epoch vs loss and epoch vs accuracy to further understand the model behaviour.

|  |  |
| --- | --- |
| Metrics | |
| Accuracy | 50% |
| Loss | 1.13 |

## 

 Initially, both losses decrease as the number of epochs increases, which is expected as the model learns from the data. However, after the third epoch, the validation loss increases while the training loss continues to decrease, which may indicate that the model is starting to overfit the training data. Overfitting happens when a model learns the details and noise in the training data to an extent that it negatively impacts the performance of the model on new data.

 The training accuracy consistently increases, which is good as it means the model is getting better at predicting the training data. However, the validation accuracy increases only slightly and then plateaus, suggesting that the model is not improving on the validation dataset as much as on the training dataset. We will now further test the performance of our model by plotting a confusion\_matrix.

## A screenshot of a computer screen Description automatically generated

## Transfer Learning Model (VGG-16)

This model provides us an accuracy of 83% which is a marked improvement from 50% accuracy of the previous model. The loss is also close to 0 which means are model is performing better.

|  |  |
| --- | --- |
| Metrics | |
| Accuracy | 83% |
| Loss | 0.29 |

## A graph of a graph showing the number of training loss and accuracy Description automatically generated

After an initial sharp decrease, the training loss levels off, which is typical as the model begins to converge to a solution.  Like the training loss, validation loss decreases sharply at first and then stabilizes. However, it appears to be slightly increasing towards the later epochs, which could be a sign of overfitting, where the model performs well on the training data but is failing to generalize to unseen data.

The training accuracy seems to be quite high and stable, indicating the model performs well on the training set. The validation accuracy appears relatively stable after the initial epochs, but it does not reach the same high level as the training accuracy, which again suggests the model may not generalize as well to new data. We have however achieved a good level of accuracy and a decreased loss, so we will further evaluate our model using a confusion matrix.

## A screenshot of a diagram Description automatically generated

## Insights & conclusion:

## The project helped me asses how we can leverage CNNs when working with limited datasets. While being able to work with larger datasets to train the neural networks is ideal, we may not always have the option to acquire more data.

## Rabbit Image Classification is challenging as the breeds have similar patterns.

## Moving forward:

## Display probability of the predicted class in the flask app

## Try data augmentation for the same dataset