# Animals classification project (CNN)

## Introduction

The objective of this report is to present the various Convolutional Neural Network (CNN) architectures explored throughout the project, including both custom and transfer learning models. It also outlines the results obtained from these experiments and discusses the initial challenges encountered during the development process.

## Methods and architecture

After the initial struggle, when the trainings were failing one after another, things finally started to stabilize. Starting from scratch was the key to get the first model which hit almost 0.7 in validation accuracy. That was real leap after several detraining epochs. After some trial and error, tweaking and documentation, we finally got something more stable.

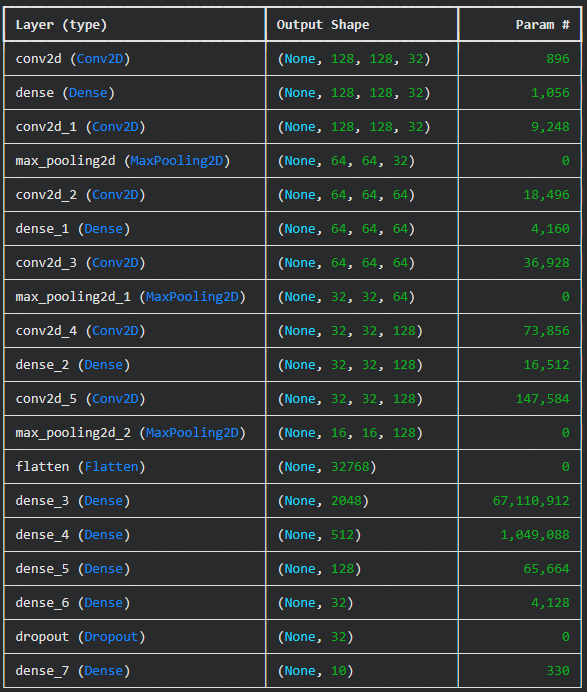
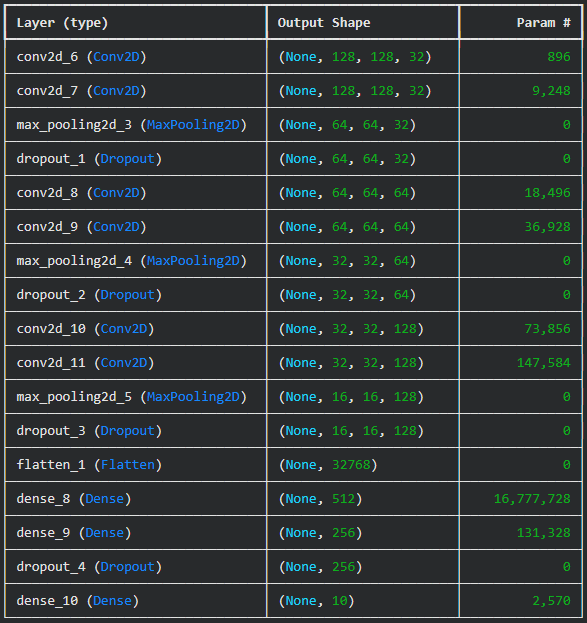
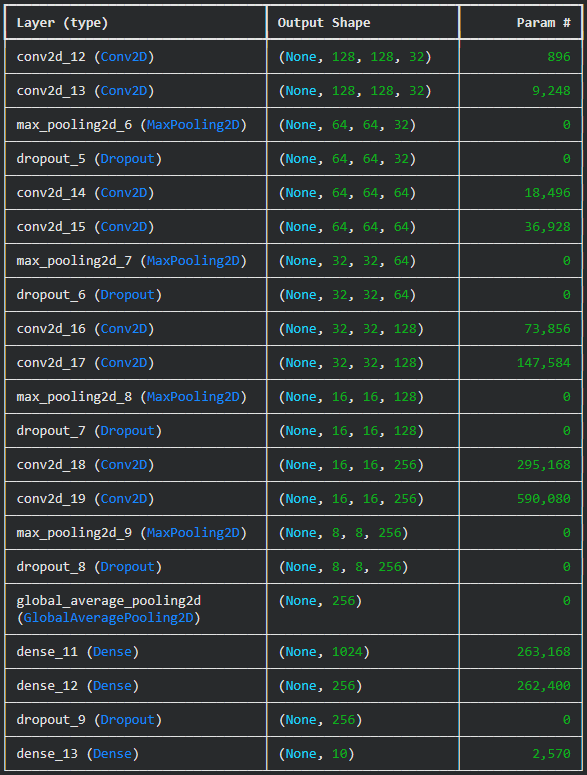
### Custom model

#### Architectures

We started with a simple model (Architecture 1) with 3 blocks of Conv2D and MaxPooling2D, adding Dense layers between them from 32 filters up to 128 and finally increasing the Dense layers to the output strongly as a final block.

In the second architecture Dense layers were removed inside de Conv2D blocks, keeping it simpler. Dropouts were added at the end of each Conv2D blocks in ascend order (from 0.1 to 0.5). The final Dense Layers were simplified as well to reduce model complexity.

The third architecture is similar to the second one but adding another double Conv2D block up to 256 filters to get more details from the images. The Flatten layer was replaced by a GlobalAveragePooling2D, allowing for better spatial information aggregation and improved generalization while reducing the total number of parameters:

From left to right (architecture 1, 2 -center- and 3 -right)

#### Model comparison

Models ummary table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Architecture | Image shape | Epochs | Learning Rate | Optimizer | Final val accuracy | Final training accuracy | Final validation loss |
| 1 | 128x128 | 35 | 0.0002 | Adam | 0.695 | 0.67 | 0.72 |
| 2 | 128x128 | 35 | 0.0002 | Adam | 0.799 | 0.816 | 0.61 |
| 3 | 128x128 | 35 | 0.0002 | Adam | 0.833 | 0.84 | 0.53 |

#### Transfer Learning

## ~~Deployment~~

Tried

## Conclusions

During the different training phases conducted with the custom CNN architectures and transfer learning models, several insights were obtained regarding model performance, optimization, and computational efficiency:

#### Input image size and performance trade-off

The use of heterogeneous image dimensions, specifically 224 × 224, proved challenging for the custom CNN model. While higher resolutions theoretically provide richer spatial information, in practice, this setup significantly increased training time and computational load. The model converged slowly and required considerable resources to process each epoch, reducing experimental efficiency.

#### Optimizer experimentation

Another optimizer was tested (AdamW), but did not produce a substantial improvement over the baseline Adam configuration. Most adjustments to the learning rate or optimizer type led to marginal or unstable gains, suggesting that the model’s performance limitations were more related to architecture and data variability than to the optimizer itself.

#### Training duration and epochs

The experiments were capped at 35 epochs due to time and resource constraints. However, the model was still improving in both training and validation accuracy near the final epochs. This indicates that additional epochs could have yielded better results (adding dynamic learning rate schedule or fine-tuning phase).

#### Fine-tuning and incremental training

Future improvements could be achieved by leveraging the *initial\_epoch* parameter in Keras. This would allow training to resume from a previously saved checkpoint, effectively enabling a second phase of fine-tuning with a reduced learning rate. This strategy would help the model refine its learned representations without losing previously acquired knowledge.

#### Impact of data augmentation

Although data augmentation techniques were applied (such as rotations, flips, and zooms), they did not lead to a significant boost in validation accuracy. This suggests that the dataset already contained a reasonable amount of variability, or that the augmentation transformations did not effectively represent the intra-class diversity of the dataset.

#### Transfer learning and hardware considerations

Transfer learning models (e.g., based on VGG16, EfficientNetB7) showed clear advantages. But… problems when predciting