Table of Contents

[Theoretical principles 2](#_Toc136217800)

[Techniques and algorithms employed. 3](#_Toc136217801)

[Exploring the data 3](#_Toc136217802)

[Creating the model 5](#_Toc136217803)

[Experiment 1 5](#_Toc136217804)

[Experiment 2 6](#_Toc136217805)

[Results analysis 9](#_Toc136217806)

[Conclusions 10](#_Toc136217807)

# Introduction

# Theoretical principles

Convolutional Networks for Image Classification:

Convolutional Neural Networks (CNNs) have emerged as a powerful variant of neural networks for image classification tasks. Unlike traditional fully connected networks, CNNs are specifically designed to leverage the spatial structure of images. They excel at capturing local patterns and hierarchies of features, making them highly effective for image analysis.

One of the key reasons why CNNs are better suited for image classification is their ability to handle the large amount of data inherent in images. Images typically have high-dimensional inputs, where each pixel represents a feature. Convolutional layers exploit the local connectivity and weight sharing to extract relevant features from different regions of the image. This approach significantly reduces the number of parameters compared to fully connected networks, making CNNs more efficient to train and better able to generalize to new, unseen images.

Additionally, CNNs employ a series of specialized layers that enhance their performance in image classification tasks. Convolutional layers use filters to convolve across the input image, capturing local patterns such as edges and textures. Pooling layers reduce the spatial dimensions of the features, promoting invariance to translation and scale variations. These layers help to extract and abstract meaningful information from the images, enabling CNNs to learn hierarchical representations that can discriminate between different classes.

Another advantage of CNNs is their ability to learn spatial hierarchies of features. Lower-level layers capture simple local features, such as edges or corners, while deeper layers combine these low-level features to detect more complex and abstract concepts, such as shapes or objects. This hierarchical feature learning allows CNNs to effectively represent the varying levels of abstraction present in images, facilitating accurate classification.

Convolutional Neural Networks have proven to be superior for image classification due to their ability to handle the spatial structure of images, parameter efficiency through weight sharing, specialized layers for feature extraction and dimensionality reduction, hierarchical feature learning, and non-linear activation functions. By leveraging these advantages, CNNs can effectively learn and classify images, making them a fundamental tool in computer vision and image analysis applications.

# Techniques and algorithms employed.

For the initial trainings and experiments, dataset

Computer upon which experiments had taken place:

* MacBook Air M1, 8GB Ram, 256GB storage

## Exploring the data

The initial dataset of 20 classes amounts to a total of 1115 instances distributed as follows:

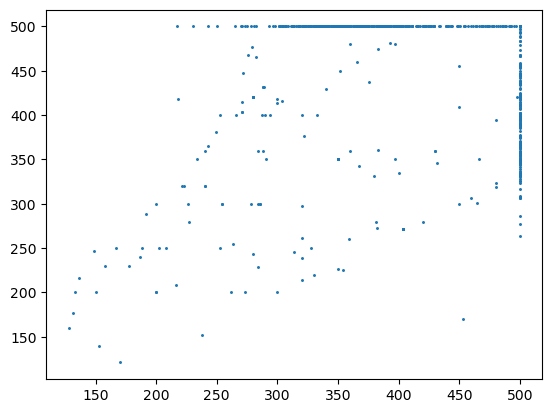
A picture containing text, screenshot, font, parallel

Description automatically generated

In here it can be observed that classes are not significatively unbalanced, worth to notice however that the Auklet family and the Spotted Catbird present around 22% less values than the rest of the classes. The information offered from the chart above, suggest that the amount of data may be not enough to preform classification tasks for 20 classes successfully, since there are approximately only around 60 instances per category.

It is important to mention that format of every image is not consistent at the time of loading and processing images initially, to solve this issue a series of transformation have been applied to the original images to get a homogeneous dataset, it’s been selected a size of 224 pixels width and height for the processed images.

Here an image of the size spread of the instances:



A quick overview to some images to get familiar with the data shows the following:

A collage of birds

Description automatically generated with low confidence

## Creating the model

For the current case, as discussed previously, will be used a Convolutional Neural Network. The initial model will have three convolutional layers, each accompanied by a ReLU activation function and a Pooling Layer, for feature extraction, and three fully connected layers to perform the classification part.

**Architecture**:

* First Convolutional Layer: 3 inputs as for the initials RGB channels, 16 outputs, and a kernel of size 5.
* Second Convolutional Layer: 16 inputs as for the outputs of previous layer, 32 outputs and a kernel of size 4.
* Third Convolutional Layer: 32 inputs as for the outputs of previous layer, 64 outputs and a kernel of size 3.
* Flattening Layer: 64 channels by image size resulting of the process of pooling(25\*25), 40000 outputs
* First Linear Layer: 40000 (64\*25\*25) inputs, as result of flattening out the output of the last convolutional layer, 120 outputs.
* Second Linear Layer: 120 inputs, as for the outputs of the previous Linear Layer, 84 outputs.
* Third Linear Layer: 84 inputs, as for the outputs of the previous Linear Layer, 20 outputs as for the number of classes.

The dataset has been partitioned in train and validation sets, to measure the performance of the model as the training is happening. In the current problem proportions have been selected as follows:

* Training set: 70%
* Validation set: 30%

### Experiment 1

Having the previous model, first experiment is conducted over the original dataset of 1115 images.

Initial parameters for the training:

* Loss Function: Cross Entropy, as the problem is multiclass classification.
* Learning rate: 0.01
* Epochs: 15

Without any other parameter definition, the model is trained showing the following results:

A picture containing text, plot, line, diagram

Description automatically generatedA picture containing text, plot, diagram, line

Description automatically generated

Execution time: 1m

Near epoch 12 it´s evident that on validation data both the loss and the accuracy start worsening, while on training data both accuracy and loss keep improving, showing signs of high variance. This overfitting problem could be associated to various conditions, in this case is believed to be related to the amount of data available or hyperparameters chosen, since the model applied is not a very complex one and classes are not especially imbalanced.

### Experiment 2

Since data available and hyperparameters are believed to be the main cause of the poor performance of the previous model, these issues will be addressed in the current one. Also noting the irregularities on the curves shown previously, it’s believed that may be related to a high learning rate. To solve the data available problem, data augmentation will be performed on the dataset, transformation operations of rotation, cropping and flipping are done.

A modified image could look like the following:

A bird sitting on a rock

Description automatically generated with low confidence

* The data set has been increased to the number of 5575 instances, the proportion between training and validation data remains the same.
* The learning rate has been reduced to 0.001.

Results of the model as follows:

A graph of training and validation

Description automatically generated with low confidenceA picture containing text, diagram, plot, line

Description automatically generated

Curves of accuracy and loss seem less irregular, and the overall performance has improved, although not to a good level yet. At the end of the las epoch, it seems the model still has room for improvement, there are no signs of high variance as validation loss keeps decreasing. To try speed up convergence it would be interesting to use momentum as an optimization method that enables the model to converge faster.

* Momentum: 0.9

The updated results show a better overall performance:

A picture containing text, screenshot, plot, diagram

Description automatically generatedA picture containing text, plot, screenshot, diagram

Description automatically generated

Important to notice that reaching the 13th epoch, validation loss starts increasing while training loss keeps decreasing, similar phenomenon with the accuracy, validation accuracy starts decreasing while training accuracy keeps improving, signs again of overfitting. To tackle these issues, this time it’s been decided to apply two layers of dropout, each dropping a 20% of links, one before the final layer and other before the flattening layer, obtaining the following result:

A picture containing text, screenshot, plot, diagram

Description automatically generatedA picture containing text, screenshot, plot, diagram

Description automatically generated

As previous images show, the result obtained is better compared with the previous ones, forecasting a better performance of the model with the hyperparameters discussed above, it was decided to also increase the size for the dataset for this last test to a total amount of 33450 instances. As expected, the running time increased by a significant amount of time, taking up to 68m.

A picture containing text, screenshot, plot, diagram

Description automatically generatedA picture containing text, plot, line, diagram

Description automatically generated

Where a noticeable improvement can be seen. As expected, the running time of the training process has increased, reaching 11m.

Time needed: 56m.

Results are significantly better than obtained in previous experiments. Achieving up to a 95% in accuracy and reducing consistently the loss down to 0.18.

A picture containing text, screenshot, plot, diagram

Description automatically generated A picture containing text, screenshot, plot, diagram

Description automatically generated

# Results analysis

# Conclusions