IMAGE CLASSIFICATION USING CNN

FOALEM Patrick Loic
Department of Computer Science
Polytechnique Montreal, Canada
`patrick-loic.foalem@polymtl.ca`
`Matricule: 2134563`

Zahra(Mina) Parham
Department of Computer Science
Polytechnique Montreal, Canada
`zahra.parham@polymtl.ca`
`Matricule: 2122841`

Athul Sreemathy Raj
Department of Computer Science
Polytechnique Montreal, Canada
`athul.s-raj@polymtl.ca`
`Matricule: 2104345`

Team
MAP3.0

ABSTRACT

The objective of the competition was to implement an efficient algorithm to classify a given set of animal images into 11 classes. We used various solutions and finally achieved an F1 score of 72%.

1 INTRODUCTION

The dataset is a bunch of animal images, grouped into 11 different classes. It contains $11,887$ images in the training set and $17,831$ in the test set. Each image has a size of (96, 96) pixels and is composed of just 1 channel. The competition was to classify the images using an appropriate algorithm that produced the best F1 score. We applied SVM, logistic regression and KNN as a baseline, and further improved our stance with Convolutional Neural Networks(CNN)(Springenberg et al., 2015).

2 FEATURE DESIGN

2.1 PREPROCESSING

The features ranged between 0-255. Hence, we scaled them down to a range of 0-1 by dividing them with 255. This is similar to a min-max scaler for the range [0,255]. This was done to make sure the model doesn't skew by the big range of values. The next step was to encode the labels. Since the labels were English words, we encoded them to numbers from 0-10, as directed in the instructions.

2.2 DATA BALANCING

On deeper analysis, we understood that the dataset contained uneven class distribution. The minimum number of examples for a class was 600 and the maximum was above 1900. Hence, the models were unable to learn more details about a few of the classes compared to the rest. To resolve this, we balanced the data by adding a selective data augmentation step. For classes that contained less than 700 examples, the dataset was tripled; for those between 700 and 1000, the sample size was doubled and the rest were left untouched. For this augmentation we used horizontal and vertical flips.

Table 1: Distribution of classes before and after balancing

Labels	*big_cats*	*butterfly*	*cat*	*chicken*	*cow*	*dog*	*elephant*	*goat*	*horse*	*spider*	*squirrel*
Before	1238	845	707	1239	746	1984	617	762	1076	1928	745
After	1238	1690	1414	1239	1492	1984	1851	1524	1076	1928	1490

2.3 Data augmentation

Further data-augmentation was applied to this balanced dataset by applying random rotation, random zoom and random brightness. We had to use a stream data generator for this since storing the augmented data in memory led to memory leaks. Hence we made use of ImageDataGenerator to create augmented images on the fly.

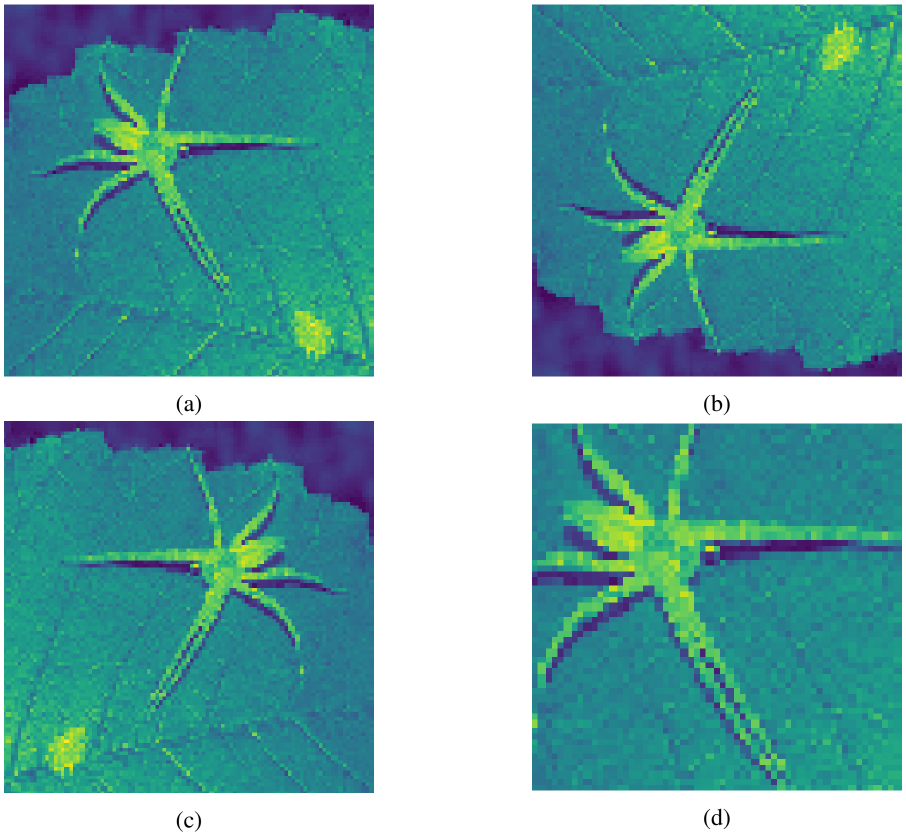


Figure 1: (a) Original, (b) Vertically flipped, (c) Horizontally flipped and (d) Scaled images after augmentation.

2.4 Data transformation

The input dataset had to be flattened to train the baseline models which only take one dimensional inputs. To train the CNNs, we had to transform the encoded labels into vectors, using OneHotEncoding, to make the output comparable to the results.

2.5 CNN Kernel selection

We tried majorly two different kernel sizes - 5x5 and 3x3 since the image size was relatively small(96x96) and the objects under consideration differed in size a lot. There were images of a litter of puppies which was very different from that of a dog's portrait. Hence, the idea was to choose smaller kernels so that we don't miss any detail. A stride of 2 was chosen, so that we don't miss any pixels and also to minimize image compression and detail loss.

3 ALGORITHM

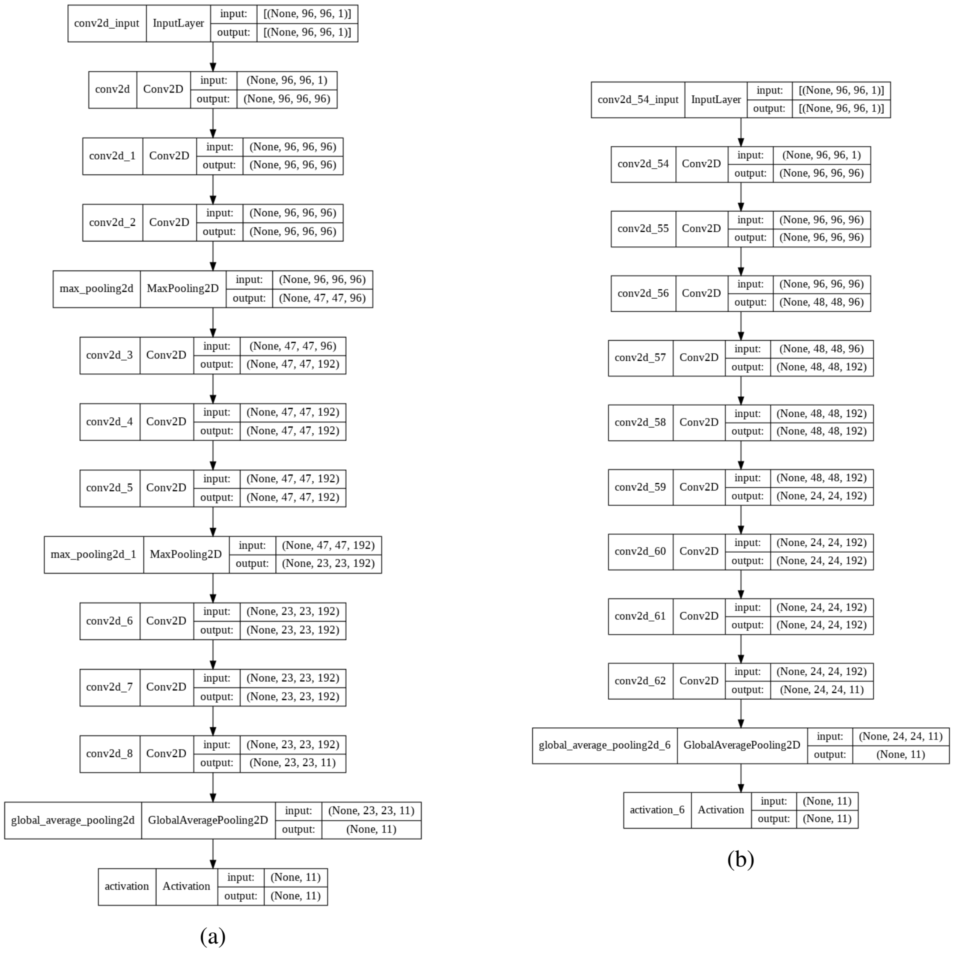
We used logistic regression, KNN, Linear SVM and as our baseline model by using sklearn libraries. Also we use the PCA algorithm to reduce the dimensionality of such datasets. PCA reduces the number of features up to 1000 which makes our training a lot faster.

1. Linear SVM is Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.(bwo)
2. Logistic Regression: LR maximizes the posterior probability; it takes C as a parameter. LR enhances predictive performance when class-conditional density assumptions give a poor approximation to the true distributions.
3. KNN: We also implemented the KNN model, which looks at the k-nearest neighbors and gives a majority of labels as the prediction. The hyperparameter for this model is K, and also it's a non-linear model
4. Convolutional neural networks ("CNN"), or multi-layer, neural networks where layers apply a convolution in the mathematical sense. A CNN consists of an input and an output layer, and multiple hidden layers. The hidden layers typically consist of convolutional, pooling, fully connected, and normalization layers. CNN's are primarily used to classify images, cluster them by similarities, and then perform object recognition. Compared to its predecessors, the main advantage of CNN is that it automatically detects the essential features without any human supervision. As the number of layers and nodes in the hidden layer increases, it tends to overfit. The probabilities of each class given an input are obtained using softmax. When making a prediction, the class with the highest output probability is chosen. We added batch-normalization to the loss function to prevent the model from overfitting. The model we used was ConvPool-CNN-C (Springenberg et al., 2015) We chose the architecture of our models by trial and error. We try many architectures and these two proved to be the best. The model architecture was composed of 10 convolution layers, followed by a linear hidden layer, and the softmax linear output layer. After each convolution layer was added a relu activation function. We use Batch normalization to normalize the batch at each convolution layer. We used strike of 3 and max pooling of 3 to decrease the dimension of the features slowly.The philosophy behind this choice of architecture was to have enough capacity to learn to discriminate complex images while also having enough regularization to prevent the model from learning the noise. The performance of this model was very better then the baseline - this was expected since it had larger capacity and it was easier to regularize with the use of dropout and batch normalization. We also used Adam with negative log-likelihood loss to train this model.
5. Ensemble Model:"In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. Unlike a statistical ensemble in statistical mechanics, which is usually infinite, a machine learning ensemble consists of only a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives."(ens) We use two models for the ensemble method: First model: ConvPool-CNN-C(Springenberg et al., 2015) Second model: ALL-CNN-C(Springenberg et al., 2015) Convolutional layers with a stride of 2 are used in place of max pooling layers. It can be seen in the structure of the second model in Appendix. Even though each of these models showed higher accuracy when trained independently, we did not get a good result with the ensemble model.

The hyperparameters of all models are in the tables below.

Table 2: Training hyperparameters of the Linear SVM

C range	best C	random state	penalty	loss
[0.0001, 0.001,0.01, 0.1, 1, 10]	0.01	0	l2	squared hinge



conv2d_input InputLayer input: [(None, 96, 96, 1)] output: [(None, 96, 96, 1)]
conv2d Conv2D input: (None, 96, 96, 1) output: (None, 96, 96, 96)
conv2d_1 Conv2D input: (None, 96, 96, 96) output: (None, 96, 96, 96)
conv2d_2 Conv2D input: (None, 96, 96, 96) output: (None, 96, 96, 96)
max_pooling2d MaxPooling2D input: (None, 96, 96, 96) output: (None, 47, 47, 96)
conv2d_3 Conv2D input: (None, 47, 47, 96) output: (None, 47, 47, 192)
conv2d_4 Conv2D input: (None, 47, 47, 192) output: (None, 47, 47, 192)
conv2d_5 Conv2D input: (None, 47, 47, 192) output: (None, 47, 47, 192)
max_pooling2d_1 MaxPooling2D input: (None, 47, 47, 192) output: (None, 23, 23, 192)
conv2d_6 Conv2D input: (None, 23, 23, 192) output: (None, 23, 23, 192)
conv2d_7 Conv2D input: (None, 23, 23, 192) output: (None, 23, 23, 192)
conv2d_8 Conv2D input: (None, 23, 23, 192) output: (None, 23, 23, 11)
global_average_pooling2d GlobalAveragePooling2D input: (None, 23, 23, 11) output: (None, 11)
activation Activation input: (None, 11) output: (None, 11)
(a)
conv2d_54_input InputLayer input: [(None, 96, 96, 1)] output: [(None, 96, 96, 1)]
conv2d_54 Conv2D input: (None, 96, 96, 1) output: (None, 96, 96, 96)
conv2d_55 Conv2D input: (None, 96, 96, 96) output: (None, 96, 96, 96)
conv2d_56 Conv2D input: (None, 96, 96, 96) output: (None, 48, 48, 96)
conv2d_57 Conv2D input: (None, 48, 48, 96) output: (None, 48, 48, 192)
conv2d_58 Conv2D input: (None, 48, 48, 192) output: (None, 48, 48, 192)
conv2d_59 Conv2D input: (None, 48, 48, 192) output: (None, 24, 24, 192)
conv2d_60 Conv2D input: (None, 24, 24, 192) output: (None, 24, 24, 192)
conv2d_61 Conv2D input: (None, 24, 24, 192) output: (None, 24, 24, 192)
conv2d_62 Conv2D input: (None, 24, 24, 192) output: (None, 24, 24, 11)
global_average_pooling2d_6 GlobalAveragePooling2D input: (None, 24, 24, 11) output: (None, 11)
activation_6 Activation input: (None, 11) output: (None, 11)
(b)

Figure 2: (a) and (b) are the CNN architectures which gave the best performance. Among them, (a) produced better accuracy and hence was chosen.

Table 3: Training hyperparameters of the KNN

K range	best K
np.arange(5,30,5)	5

Table 4: Training hyperparameters of the Logistic regression

C range	best C	max iter	random state	penalty
[0.0001, 0.001,0.01, 0.1, 1, 10]	0.001	10000	0	l2

Table 5: Training hyperparameters of the CNN

learning rate	momentum	weight decay	epochs	batch size
0.0001	0.9	0	100	32

4 METHODOLOGY

We used a train/valid split of 0.8/0.2 for all the algorithms that we have implemented (LR, SVM, KNN, CNN).

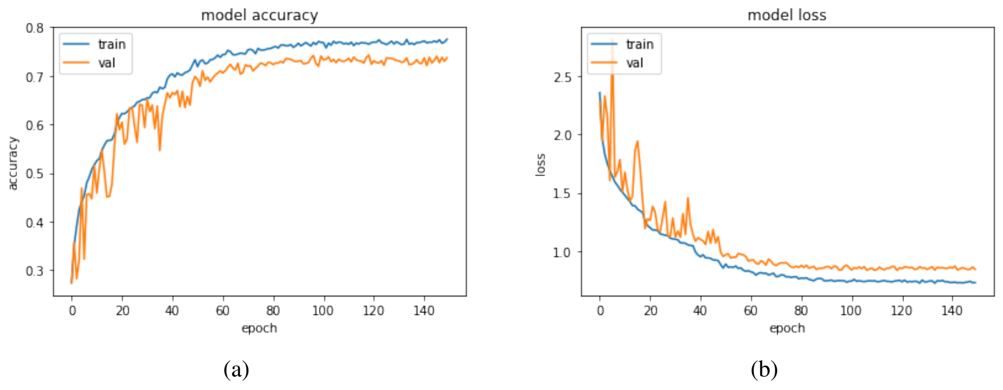


Figure 3: (a) Accuracy and (b) Loss of the classifier throughout epochs. One unit on the horizontal axis is equivalent to 10 epochs.

1. Logistic Regression : We tested the multinomial logistic regression with L2 regularisation. We perform some hyperparameter optimization as mentioned in section 3. We tested this algorithm with data augmentation and as we expected the model didn't perform well.
2. Support Vector Machine : We tested the linear version of SVM for multiclass classification with some hyperparameter optimization as mentioned in section 3.
3. k-nearest neighbors : We tested KNN as our baseline with some range of K with the same data augmentation technique.
4. Convolutional Neural Network : We tested 5 different CNN architectures, for each architecture we tried 3 different optimizers (Adam, SGD, rmsprop) at the end we considered Adam as our optimizer since Adam showed the best accuracy. The selected architecture is described in section 3. All architectures were tested with the same preprocessing and data augmentation. The model that performed well in the validation set was saved and used for the ensemble model.

5 Results

Accuracies for all models are shown in table 6. The CNN classifier was generalizing a lot better than other classifiers.

Table 6: Validation and train accuracy of the selected CNN classifier and the baseline.

model	training	validation
CNN	75.21	71.79
logistic regression	24.5	24.0
support vector machine	20.91	20.1
k-nearest neighbors	11.21	23.4

6 Discussion

The performance of the CNN model was much better that the performance of other model. This result was expected since the CNN model has more capacity then logistic regression, support vector machine, k-nearest neighbors and the CNN is more appropriate for image related tasks because of its filtering mechanism. We achieved relatively good accuracy. We used the Google collab GPU for a fast execution of our CNN model, to improve our accuracy we used the LearningRateScheduler, ModelCheckpoint, ReduceLROnPlateau functions as Callbacks parameters to monitor the training process at each epoch.

7 Statement of Contribution

The problem was discussed in detail among the three of us and we took the time to split up tasks. Patrick took the lead to initiate meetings and made sure we meet frequently. Athul, being experienced in the industry, set up the boiler-plate code, and completed the pipeline in a modular format. Patrick, being a newbie, explored literature and blogs about improving the performance. Mina, who was already a machine learning engineer, took the steps to explore advanced ML models including CNNs. Once the preprocessing and baseline was set by Athul, Patrick came up with a CNN architecture which improved the model accuracy to more than 60% which was a huge leap. Mina tried ensemble methods with multiple CNN algorithms to produce better performance. Athul came up with the idea of data balancing and implemented it. Afterwards, Mina, Athul, and Patrick took turns in training multiple CNN architectures to improve the existing performance since the number of submissions were limited. This greatly improved their knowledge in applying ML to solve problems. Like the rest of the project, report was also split into parts, and was done with equal contribution from all the three participants.

We hereby state that all the work presented in this report is that of the authors.

References

Support Vector Machine Algorithm. `https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm`.

Ensemble learning. `https://en.wikipedia.org/wiki/Ensemble-learning`.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net, 2015.