KU LEUVEN

FACULTY ENGINEERING TECHNOLOGY
CAMPUS GROEP T

Machine Learning: project

Emotion detection of pets

Using fundamentals of machine learning to recognize four emotions of cats and dog.

r0819932 - J. Hsiung
r0837926 - J. Aerts

Coaches:
Meixing
Meng

Deadline: mon 27 november '23 midnight

Table of contents

Introduction

For our course Machine Learning (ML) we were challenged to find a problem and tackle it to show to power of fundamental ML techniques like regression, Convolutional Neural Networks (CNN), and more. In this rapport we will show our first milestone. Firstly, we will talk about the motivation of this project and problem we're going to tackle. Secondly, we'll propose a solution to this problem. In the third section we will provide a small section on the definition of used techniques. After that, we'll provide some definitions used in this project. In the Fourth section we will be talking about how we aimed to solve the problem. In the fourth and the fifth section we will go in more detail about the data analysis, approaches, methods, software, designs, and more. Following up in the sixth section we will show our preliminary result and compare them with similar projects found on different platforms like GitHub and Kaggle. In the last section we will talk about how we can improve our first model and motivate by using the knowledge we gained from the theory given by prof. B. Vanrumste and self-gained knowledge from extracurricular efforts.

1 Motivation

In Extended Reality (XR), the integration of pets into virtual spaces has created for a range of new possibilities. However, with the discoveries of new technologies, there is an inherent void of unexplored domains. In this project we aim to enhance the XR technology by developing a model that a) identifies cats and dogs in a picture and b) detects the emotion of the depicted cat or dog – happy, angry, sad, or relaxed.

Use case.

An application of our model could be found in remote pet monitoring for healthcare assessment. By using trail cameras biologist, research, veterinaries, or even hobbyist could remotely observe the emotional state of an animal; conclude distress; observe behavioral change and more. We believe this model has valuable potential when extended to different species and various emotions.

2 Solution

Our model is a CNN regression-based model. The CNN models have shown their success across the broad relevant literature in image classification to capture feature vectors as explained in the ML course provided by the Faculty of Engineering Technology at the Katholieke Universiteit Leuven (KU Leuven).

Using best practices learnt from the courses as well as best practices from developer fora, we developed a custom CNN architecture with added convolutional layers and pool layers for feature extraction and spatial reduction respectively. After our work we finally got a model that is abstractly depicted in figure 1.

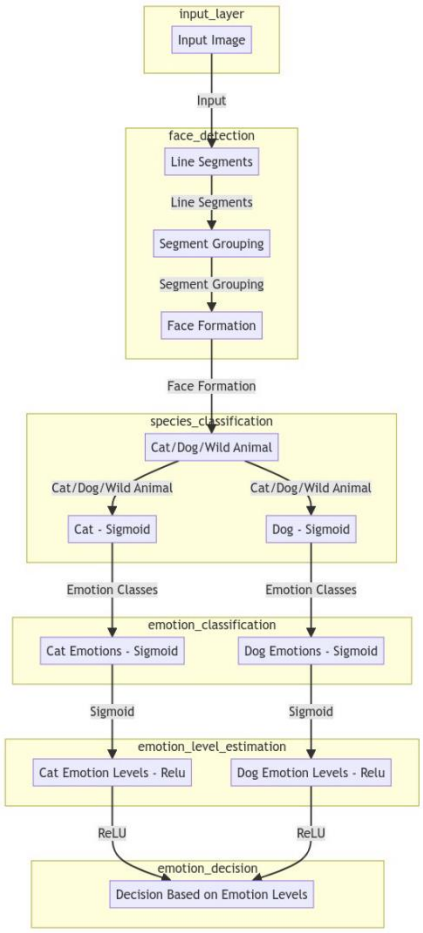

Figure 1: *Abstracted depiction of model for emotion detection of cats and dogs*

3 Definitions

// Will be added in final report

4 Dataset and Features

One of the requirements of this project is that we use a base model. We opted for a simple neural network with self-defined logistic regression, linear regression, regularized logistic regression, regularized linear regression, sigmoid activation function, linear activation function, Rectified Linear Unit (ReLu) activation function.

Our dataset includes a variety of animals of different species with different emotion, we used RGB-valued png-images, which can include one or two animals at the same time. These were imported from various sources like GitHub, Kaggle, and were both manually and pre-labelled. We used RGB-valued images because we found that for animal recognition, we had a higher accuracy when using RGB-valued images versus grey scaled images.

These images are then transferred into useful tensor image data using the `ImageDataGenerator` from the `tf.keras.preprocessing.image`[1] library package. This generator "Generate batches of tensor image data with real-time data augmentation.". This is useful because of the data augmentation aspect of it.

> "Data augmentation is a technique of artificially increasing the training set by creating modified copies of a dataset using existing data. It includes making minor changes to the dataset or using deep learning to generate new data points." - datacamp[2]

5 Methods

We approached this problem using the following reasoning. First, we need preprocess the data. Next, we to detect whether the image contains a dog and/or a cat. Then we need to classify which emotion the animal in the depicted images has. Followed by some optimization.

5.1 Animal classification

We used the same approach as a conventional image recognition model would as seen in figure 2, where the first hidden layer generates line segments. The second hidden layer generates grouped segments, which essentially are parts of the face, and the last hidden layer generates faces.

Since we want the first output (which, still, is a hidden layer) to contain the binary classification of dog and cat, we want to have a feature vector that looks like the code snippet 1, hence we will use a sigmoid activation function at the output layer.

```
feature_vector_animal_detection = [input_contains_a_dog,
input_contains_a_cat]
```

Code snippet 1: *feature vector containing necessary data for cat/dog classification.*

5.2 Emotion classification

In the subsequent layers, our goal is to obtain an output feature vector as depicted in code snippet 2. Again, the elements of this feature vectors are Boolean containing either zero or one. Considering that the input image may contain either a cat, a dog, or both,

[1] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[2] https://www.datacamp.com/tutorial/complete-guide-data-augmentation#:~:text=Data%20augmentation%20is%20a%20technique%20of%20artificially%20increasing%20the%20training%20set%20by%20creating%20modified%20copies%20of%20a%20dataset%20using%20existing%20data.%20It%20includes%20making%20minor%20changes%20to%20the%20dataset%20or%20using%20deep%20learning%20to%20generate%20new%20data%20points.%C2%A0%C2%A0

the corresponding 'class' indicators[3] should be set to one or zero, indicating the presence or absence of the respective class in the input.

```
feature_vector_emotion_detection = [is_cat_happy, is_cat_sad,
is_cat_angry, is_cat_relaxed, is_dog_happy, is_dog_sad, is_dog_angry,
is_dog_relaxed]
```

Code snippet 2: *feature vector containing necessary data for emation classification.*

5.3 Multi-class input

When the input both contains a dog and a cat then an emotion category is likely. The next layer learns emotion levels for each class. For example, if an image has both a cat and a dog with detected happiness and relaxation for the cat, two neurons compute those levels. The same applies to the dog. The ReLu function (e.g., y = [cat happy, dog happy, cat sad, dog sad, cat angry, dog angry, cat relaxed, dog relaxed]), is used to represent these levels.

5.4 Output layer

This layer identifies emotional classes based on input levels. For example, if the input is `[is_cat_happy, is_cat_sad, is_cat_angry, is_cat_relaxed, is_dog_happy, is_dog_sad, is_dog_angry, is_dog_relaxed]` and the image has both cat and dog, with cat being happy and relaxed and dog being happy and sad, the input becomes `[0.5, 0.4, 0.0, 0.5, 0.0, 0.0, 0.7, 0.0]`. The layer's output should have only two ones, like `[0, 0, 0, 1, 0, 0, 1, 0]`, indicating that the cat is relaxed, and the dog is sad. The layer can use SoftMax or, for each unit, a sigmoid function. The reasoning behind used the SoftMax function inside the layer and not with the loss functions is, that we found convergence was slower when we put the SoftMax function together with the loss function.

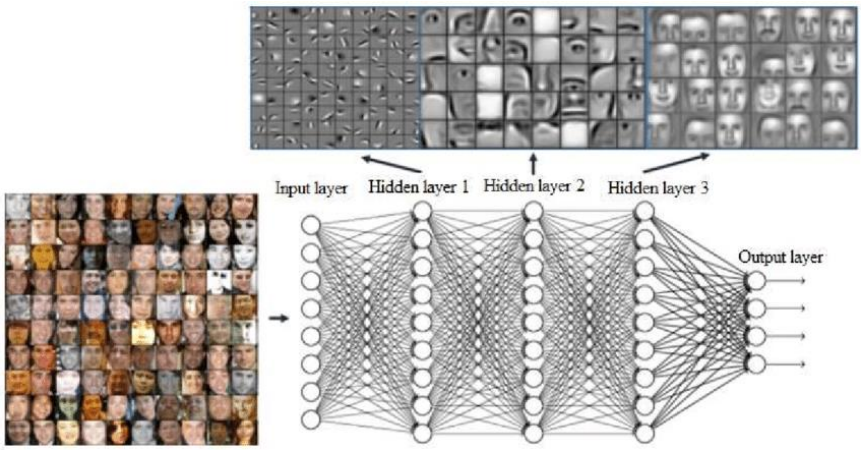

Figure 2. Features obtained for face recognition application in hidden layers of CNN.[4]

[3] With class indicators we mean a group of elements. In this example a class called 'cat' would be the collection of [is_cat_happy, is_cat_sad, is_cat_angry, is_cat_relaxed].

[4] Face Mask Detection on LabVIEW - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Features-obtained-for-face-recognition-application-in-hidden-layers-of-CNN-9_fig1_355228583 [accessed 26 Nov, 2023]

6 Preliminary results

We developed three models. The first model serves as our base model, and its structure is depicted in code snippet 3, which is found in the appendix. The second model is a combination of our base model and another model we discovered on Kaggle, as illustrated in code snippet 4. Later, we began experimenting with models not covered in the course and 'stumbled' upon the EfficientNet CNN architecture. Considering various metrics such as accuracy, training time, etc., we opted for model version B5 as our base model. One of our decision visualizations for metrics is shown in Figure 3. Using the EfficientNetvB5 as base model integrated with our model, achieved an accuracy of 90% (depicted in code snippet 5).

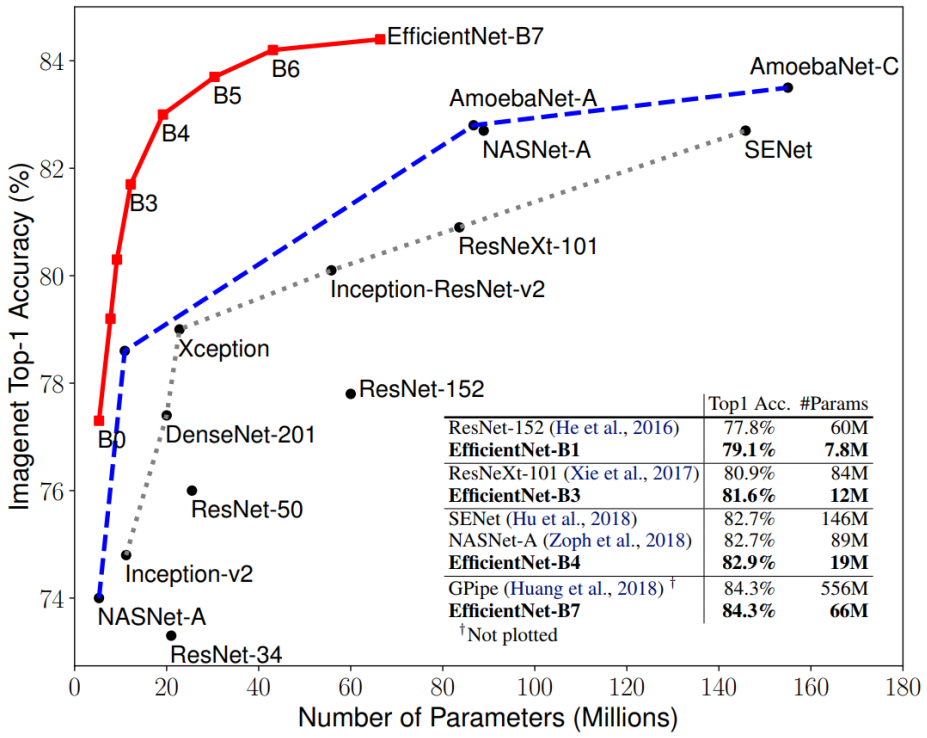

Figure 3. *EfficientNet size and performance on the ImageNet dataset (Source: image from the original paper)*

Notice how we didn't place the SoftMax function together with the loss function – whereas the lab suggested we do put it in with the loss function. The reasoning behind this choice is that we found the convergence was slower [JEFFEE HOW MUCH SLOWER?].

7 References (APA)

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

8 Appendix

```
emotion_model = Sequential([
    Flatten(input_shape=(128, 128)),
    Dense(512, activation='relu'),
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),
    Dense(3, activation='softmax')
])

emotion_model.compile(
    loss=CategoricalCrossentropy(from_logits=False),
    optimizer=Adam(learning_rate=0.001),
    metrics=[CategoricalAccuracy()]
)
```

Code snippet 3: *Base model architecture*

```
emotion_model = Sequential()

emotion_model.add(Conv2D(64, (3, 3), activation='relu'))
emotion_model.add(BatchNormalization())
emotion_model.add(Conv2D(64, (3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(128, (3, 3), activation='relu'))
emotion_model.add(BatchNormalization())
emotion_model.add(Conv2D(128, (3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(512, (3, 3), activation='relu'))
emotion_model.add(BatchNormalization())
emotion_model.add(Conv2D(512, (3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Flatten())
emotion_model.add(Dense(512, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(3, activation='softmax'))

print(emotion_model.summary())
emotion_model.compile(loss='categorical_crossentropy', optimizer="Adam",
metrics=['accuracy'])

```

Code snippet 4: *Model Kaggle*

```
 1. base_model = tf.keras.applications.efficientnet.EfficientNetB5(include_top=
False, weights= "imagenet", input_shape= img_shape, pooling= 'max')
 2. base_model.trainable = False
 3.
 4. model = Sequential([
 5.     base_model,
 6.     BatchNormalization(axis= -1, momentum= 0.99, epsilon= 0.001),
 7.     Dense(256, activation='relu'),
 8.     Dense(128, kernel_regularizer= regularizers.l2(l= 0.016),
activity_regularizer= regularizers.l1(0.006),
 9.                 bias_regularizer= regularizers.l1(0.006), activation= 'relu'),
10.     Dropout(rate= 0.45, seed= 123),
11.     Dense(class_count, activation= 'softmax')
12. ])
14. model.compile(Adamax(learning_rate= 0.001), loss= 'categorical_crossentropy',
metrics= ['accuracy'])
15.
```

Code snippet 5: *integration EffiecientNetvB5*

FIIW GROEP T
Andreas Vesaliusstraat 13
3000 Leuven, België
tel. + 32 16 30 10 30
fet.groupt@kuleuven.be
www.iiw.kuleuven.be