Dog-breed Classifier

Simon Andres Soto Ochoa

June 16, 2021

Abstract

This is the Capstone Project for the end of the Machine Learning Engineer Nanodegree from Udacity. The prject consisted in the creation from scratch of a dog breed classifier, to observe the complexity of the topic and also the use of a pre trained oject.

1 Introduction

Computer vision is a fascinating area of Computer Science as it helps humans to do jobs at an incredible fast pace. This project in particular is going to work with classification. This is a task that a human can perform but at a slower pace and constraints. In particular, classification of a dog or not, or human or not is not a complicated task for humans but it is more complicated for computers as they only perceive a collection of data specifying the RGB encoding of each pixel. Even though computers have that limitations, there are available several classifiers online to do that in a very accurate way. The project goal is to create the next step, a dog-breed classifier. This is not a trivial task, as there are many several breed that look quite alike. For a human it requires a lot of time just to memorize the name of the breed, something that a computer can do better than us, but still, after learning names, the process of choosing correctly a specific breed is another story. This has been a topic in research, for example [LKJB12] , tries to overcome difficulties by identifying more features for a breed in the same pictures. Extracting more features from the image has shown results in the case of flowers [NZ08]. There are attempts only in the area of feature extraction to find missing dogs [LTY19].

Besides this topic there are more interesting projects to conduct, but as this project goes well, will give a model from which I can base other personal projects.

In this capstone project, you will leverage what you've learned throughout the Nanodegree program to solve a problem of your choice by applying machine learning algorithms and techniques. You will first define the problem you want to solve and investigate potential solutions and performance metrics. Next, you will analyze the problem through visualizations and data exploration to have a better understanding of what algorithms and features are appropriate for solving it. You will then implement your algorithms and metrics of choice, documenting the preprocessing, refinement, and postprocessing steps along the way. Afterward, you will collect results about the performance of the models used, visualize significant quantities, and validate/justify these values. Finally, you will construct conclusions about your results, and discuss whether your implementation adequately solves the problem.

define Analyse implement results conclusions

2 Problem Definition

Classification of dog breeds is a complex topic even for humans. Therefore, we are going to construct an algorithm that can help in that process. Given a picture, determine if the picture contains a human of a dog. In the later case, determine a dog breed, and in the other case, determine the breed that the human resembles.

Restrictions: Picture must have ONLY a dog or a human. First part consists in identifying if there is a dog or a human, and the second part is to return the most likely breed or the breed that the human resemble the most. If both or none of the previous ones is found in the picture, we return an error, and ask for a new image.

Ideas of Solution: Use *Transfer Learning* from a Convolutional Neural Network (CNN) to be able to construct (not from scracht) without training our own architecture a dog classifier. We are going to define the architecture of the network but we will use a pre-trained model.

3 Problem Analysis

The data sets are publicly available. Here you can download the .zip file for the dogs pictures and here for the .zip file containing the pictures of humans.

The dog folder contains the train, validation, and test sets already in different folders in which each folder has a respective dog breed. This follows the regular standards on image classification in which each example has to be well referenced. Similar for humans, but in this case, we have just a folder named after one person and inside at least one picture of him/her. There are 8351 images of dogs and 13233 images of humans.

In our case, we have 133 different breeds (The number is the same for train, validation and test) a pre-trained model (VGG-16) for dog classification for the competition of ImageNet, and a model to identify human faces in pictures from OpenCV.

Basic figures over the amount of dogs pictures for each breed can be seen in 1.

As we saw in Section 3, for each breed, we have pictures in the train, validation, and test sets. The classes are no balanced but also not completely imbalance as it can be appreciated in figure 1. There is a difference among the top 4 breeds with more images and the bottom 4, but this does not make up for a significant difference. We decided during the process of executing the project to focus on accuracy instead of recall (as we stated in the proposal).

3.1 Justification

There are several papers [KRSB18, LKF10, BTKK19, KK+20] among others, that let us think that the best approach to this problem is the use of convolutional neural networks. Also, for the biggest competition on image recognition (ImageNet) the best models use all convolutional layers and also a sort of interesting combinations not that

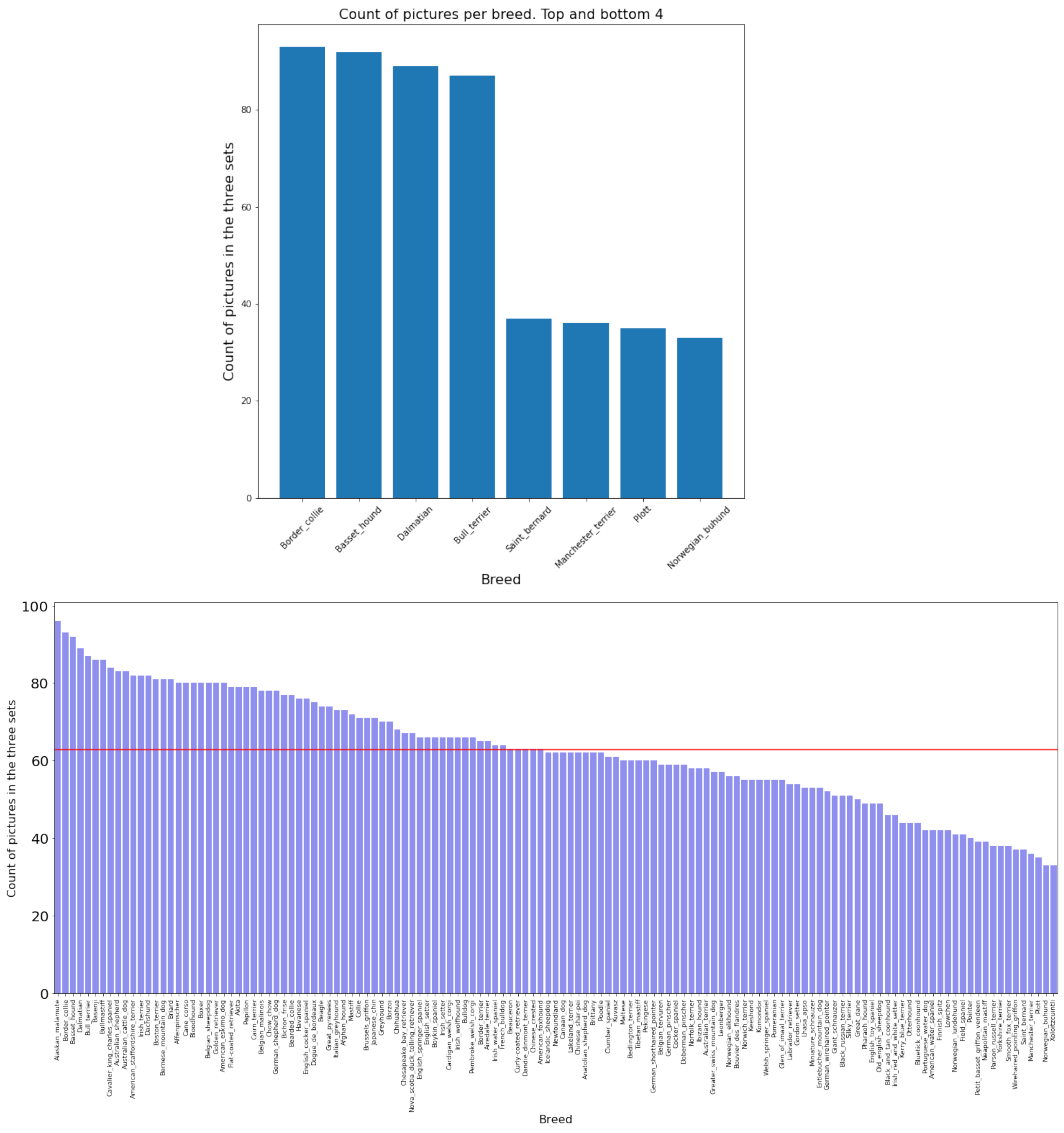


Figure 1: Number of pictures per breed adding up the three sets.

easy to emulate. The network created from scratch was built up following the basic structure from the VGG-16 model.

3.2 First Steps

3.2.1 Human Face Classification

During the start of the process, we use the OpenCV Haarcascade Frontal Face Classifier. With this one we can achieve a 99% accuracy with the first 100 pictures of faces and only mistakenly corrected 17% of the dog pictures. So, it is good if we have a human face, but not the best for a dog face.

3.2.2 Dog Classification

For the dog pictures, we have first a pre processing of them in which we flipped horizontally, rotated , and resized. This is our stressed scenario and we got a 74% accuracy with VGG16 for dog classification, and only 1% of humans misclassified as dogs. Without stress, the accuracy can jump up to 99%. We decided later to try a ResNet 50 pre trained model. With this one, under the stressed conditions, we were able to achieve a 86% of accuracy. The last let us conclude that for the second part of the project, this model would perform quite well.

4 Implementation

4.1 Algorithms and Techniques

A convolutional neural network is a kind of Deep Neural Network. The main feature of this architecture is that it is good on finding details in the pictures. The weakness of this kind of layers is that the training process is quite slow as it has many parameters to configure.

An image with dimension $w \times h$ (width times height) is an array (matrix) of numbers. In the case of a gray-scale image the deepness of the image is one, as every pixel can be represented as a number between zero and one. In the case of RGB pictures, our case, there is a layer (level) for each of the colors. Therefore, an RGB image has dimensions $w \times h \times 3$.

A textbfconvolutional layer what is trying to do is to get information from the combination on the layers into new layers. These layers are the most used way to work in computer vision and the idea behind it is the following.To make all calculation more simple, we are going to work with square images ($w = h$). What you have is an image of dimensions $w \times w \times 3$. A filter is a small sliding window of size (for simplicity, square too) $s \times s$ that is going to move through the picture and grab the details of the pixels that it contains. There are two parameters: **slide** and **padding**. The first one is how many pixels are you going to move your window to get a new entry. The second is how many pixels are you going to put outside of the original picture so the algorithm can identify the edges of the picture (which can be relevant for the application)

A filter is going to create a new layer with dimensions

$$\lfloor \frac{w + 2p - s}{s} + 1 \rfloor \times \lfloor \frac{w + 2p - s}{s} + 1 \rfloor.$$

If k is the number of filters that you want to do, then, you end up with a multi-dimensional array of dimensions

$$\lfloor \frac{w+2p-s}{s} + 1 \rfloor \times \lfloor \frac{w+2p-s}{s} + 1 \rfloor \times k$$

In this way, if each filter focuses in extracting one feature, we have at the end k different features in our new array that we can keep exploring. It is called a convolution as it let us mix different aspects present in the picture (RGB encoding).

Regarding the size of the filters, I can only mentioned that in the paper of the VGG-16 model [SZ14], the recommendation is to use 3×3 as the size of the filter. This, they say, is to identify more details on each of the layers.

An important step on all the algorithms is the **MaxPool layer**. In this layer, again, we have a filter, and the idea is to extract aggregate information about each layer. The filter will find an average value on the values of the units, and create a unique element. This process do not alter the number of filters that we started with. We have again a **slide** and **padding** that might change the dimensions of the layer.

Third element, the **activation function**. We chose ReLU that is defined as

$$\text{ReLU}(x) = \max(0, x)$$

In the literature, this function is better than the sigmoid function in terms of computational complexity, which simplifies the process of training. But there it is still a personal choice. You can also check the tanh function.

A good reference to understand these two is given in the link of the Course on Convolutional Neural Networks for Computer Vision

Another elements that we used are:

Batch Normalization: With this, we try to reduce the noise in the output and make the process more computationally simple for the network. There is still not clear reason but it has been used in the industry recently.

Fully Connected Layer: This is the most common and traditional layer that is used for classification. It consisted of neurons, excellent start in the topic is the following video made by the good channel of Three brown one blue The basic elements are neurons, that can be think as units with a value in it. Being fully connected means that every unit would have a connection to every unit in the next layer.

4.2 Preprocessing

We did two pre processing stages in the development of this project. The first one, was to classify picture with dogs or without dogs. For this step, we created two scenarios to test the power of the models.

Stressed Scenario: In this scenario we did the operation shown in the next snippet of code.

```
preprocess = transforms.Compose([
        transforms.RandomResizedCrop(size = 224), #Resize the picture
                                                   as VGG accepts only
                                                   224x224x3 y design.
        transforms.RandomHorizontalFlip(p=0.6), #Perform a horizontal
                                                   flip with a probability of
                                                   60%
        transforms.RandomRotation(degrees = 45), #Perform a rotation
                                               of the picture 45 degrees
        transforms.ToTensor(), # Transform it to a tensor.
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0
                                               .224, 0.225]) #Normalize
                                               the values.
    ])
```

The elements in that are the following: Both models, VGG16 and ResNet50 have 224×224 pixels as the entry size for the first layer. Therefore, always the first step in this stage is to reshape the pictures to fit that format. For the training images, we used a *random resized crop* in which we have 224 as our desired size. This method makes a random crop of size 224 to the image it receives. After that, a *random horizontal flip* with a probability of 0.6 to flip using the vertical axis the image. Later, a *random rotation* of 45 degrees. We concluded with a transformation to a tensor (that is the basic input of the network) and normalizing the values to make the algorithm more resilient against outliers.

Non-stressed Scenario: For the non-stressed scenario, we did not flip nor rotate the pictures before passing them to the model.

The second part was for the training of our model created from scratch and the transfer learning model. We took the same transformation that we use before with some minimum changes: First change, the probability of a flip is 0.5. Second change, the rotation is 15 degrees, but this transformation is made with a probability also of 0.5. This will challenge the training without making it too difficult to learn from.

4.3 Transfer Learning

In the exploration of a model to identify dogs from pictures we came across different models that performed in the ImageNet competition in a competitive way with respect to the VGG16. In that search we found out that Inception V3 and ResNet50 were a pair of models that outperform with the conditions given in the stressed pre processing described in 4.2. In the final decision, we went with ResNet50 because the same preprocessing (the same size of input image) can be used. For the Inception V3, we needed to change the size of the input image.

The main change while using a pre-trained model for transfer learning, is to change the final output to match your requirements. In this case, for the ResNet50 we had to change the output size from 1000 to 133 dog breeds. This can be by adding a new fully connected layer

5 Results

5.1 Benchmark model

The benchmark model is going to be a trained neural network design by us. In this one, we are going to take ideas from different sources, for example [LKJB12], which is one of the first papers that classifies the 133 dog breeds (same number as in this project). In this paper, they use the recall (True Positive Rate) to measure the performance of the model. In our case, we decided to change this measure as it was already implemented and recommended in the notebook. In this case, we use the model that we created as a benchmark for our results. This is clearly a way to also show that there is better tools outside in the market than the ones you can do with simple elements, but still that you are able to construct a Network with the pretty decent results.

The network has the following components:

- (conv1): Conv2d(3, 16, kernel size=(3, 3), stride=(1, 1), padding=(1, 1))
- (conv2): Conv2d(16, 32, kernel size=(3, 3), stride=(1, 1), padding=(1, 1))
- (conv3): Conv2d(32, 64, kernel size=(3, 3), stride=(1, 1), padding=(1, 1))
- (conv4): Conv2d(64, 128, kernel size=(3, 3), stride=(1, 1), padding=(1, 1))
- (conv5): Conv2d(128, 256, kernel size=(3, 3), stride=(1, 1), padding=(1, 1))
- (fc1): Linear(in features=12544, out features=512, bias=True)
- (fc2): Linear(in features=512, out features=133, bias=True)
- (dropout): Dropout(p=0.2)
- (pool): MaxPool2d(kernel size=2, stride=2, padding=0, dilation=1, ceil mode=False)
- (batch norm): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track running stats=True)

The first 5 iterations consisted of a convolutional layer followed by a maxpool layer and ended up with an activation with the ReLU function.

When the five convolutional layers are passed, we applied a batch normalization, and then the first fully connected layer, followed by a ReLU activation, lastly we pass to the second fully connected layer from where we get our final output.

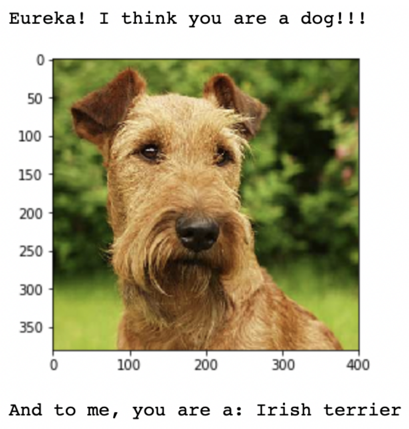
After 30+ epochs, we were able to achieve a 13% of accuracy. This can imply that our architecture needed more training (but that can lead to overfitting) or we needed a better architecture, but that would make it too complex to follow or to explain.

5.2 Transfer Learning Results

The ResNet model presented better results in the first stage and that is why it was also used to the transfer learning part of this task.

We decided based on the results that as the ResNet50 network is good at identifying not dogs (the errors in the short list was 0%) we first are going to check if there is a dog in the picture or not. With a negative answer we then ask if there is a human face in the picture. Lastly if none of those is satisfied, we get to the conclusion that there is a mistake with the picture.

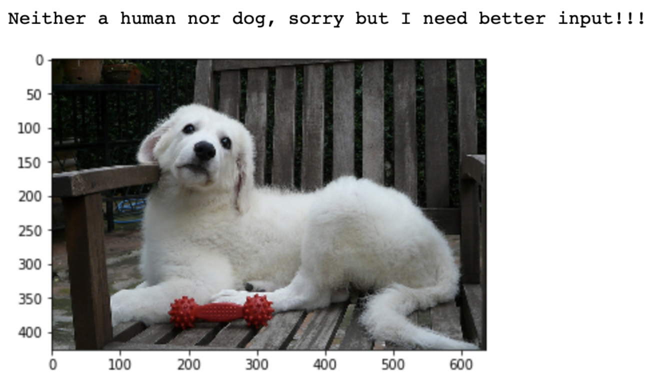
This project started as a way to learn more about computer vision, in particular, Convolutional Neural Networks. To achieved this goal, we first used known algorithms to have a human and a dog classifier. After that first step, the real work started. We developed a simple Convolutional Neural Network to find the dog breed of different dog pictures, it was trained under the ecosystem in Udacity as the GPU mode help with



```
Eureka! I think you are a dog!!!
```

```
And to me, you are a: Irish terrier
```

Figure 2: A well identified Irish Terrier



```
Neither a human nor dog, sorry but I need better input!!!
```

Figure 3: Problems that always appear

the speed at training, and we achieve a decent 13% accuracy. Some results of the app can be check in the Figures 2 and 3.

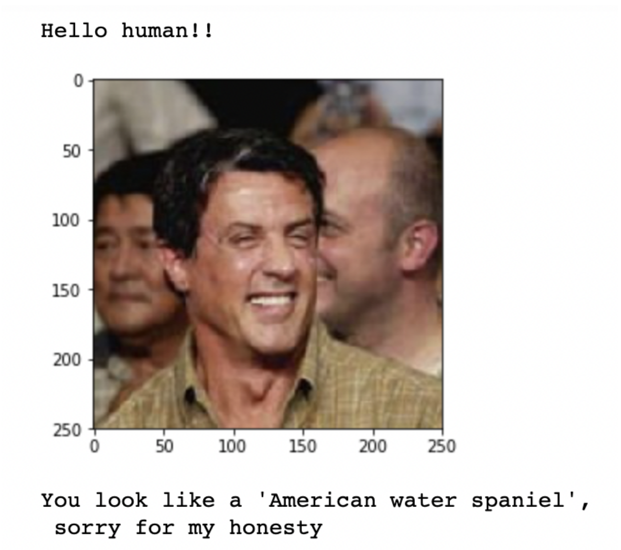
In the case for humans, identifying humans in pictures was not the most complicated task, with OpenCV was doable, and you can find funny comparisons made by the Network for the different people. You can see a sample of the work in the Figures 4,5.

Our benchmark model achieved a 13% accuracy, while in the transfer learning model we achieved an amazing 84% accuracy! This is incredible and shows the power of transfer learning in a clear way! The ResNet50 model is clearly more complex than the scratch model. This justifies the use of transfer learning to do other tasks in a related field. The accuracy of the transfer learned model gave us the satisfaction of making an amazing job.

6 Conclusions, Ideas and Future work

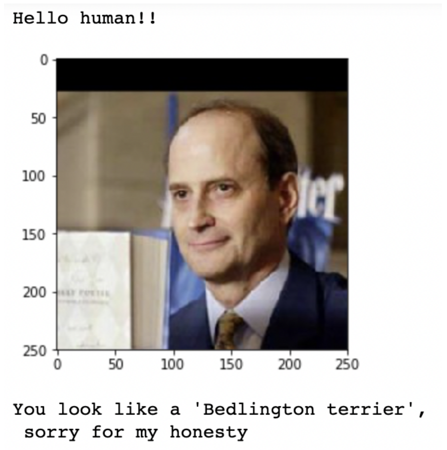
- Creating a model that satisfy the desired requirements is not an easy task and it requires a lot of time to train and to test the different ideas you might have in your head.

- Differentiating objects is an easy task for the different models that are already produced. In the case of the pre trained models, they were able to differentiate


```
Hello human!!
```

```
You look like a 'American water spaniel',
 sorry for my honesty
```

Figure 4: Sylvester Stallone as an American water spaniel


```
Hello human!!
```

```
You look like a 'Bedlington terrier',
 sorry for my honesty
```

Figure 5: A cute guess. A Bedlington terrier

with high accuracy between humans and dogs. Creating a dog breed classifier is a complete different story, there are many small features that cannot been depicted in the pictures. As mentioned in the notebook, even for humans it is a complicated task.

- We can give more training epochs to the algorithm so it might perform better. But there is a chance to over-fitting.
- The state of the art algorithms that are currently available in the market make it easy to construct your own dog breed classifier but their architecture is way more complex. This makes it difficult to follow their ideas. Even more, the complexity also translates into more training time, VGG-16 needed one week to do its training.
- Transfer learning is an interesting field that can be explore more in the future, this will let you not to do everything from scratch but to use the already built power,

made by people doing research in those areas, for our own purposes. Obviously there should be a domain investigation to use the best models for the tasks that we are interested at the moment.

- The human eyes can construct in the mind 3D models of what we see, and this deepness is something that I cannot imagine that a computer can do.
- Development of an app for smartphones in which you can take the picture and get a guess about the breed of that dog.

References

[BTKK19] Punyanuch Borwarnginn, Kittikhun Thongkanchorn, Sarattha Kanchanapreechakorn, and Worapan Kusakunniran. Breakthrough conventional based approach for dog breed classification using cnn with transfer learning. In *2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–5. IEEE, 2019.

[KK$^+$20] Aman Kumar, Amrit Kumar, et al. Dog breed classifier for facial recognition using convolutional neural networks. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 508–513. IEEE, 2020.

[KRSB18] Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, and Mohammed Bennamoun. A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8(1):1–207, 2018.

[LKF10] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[LKJB12] Jiongxin Liu, Angjoo Kanazawa, David Jacobs, and Peter Belhumeur. Dog breed classification using part localization. In *European conference on computer vision*, pages 172–185. Springer, 2012.

[LTY19] Kenneth Lai, Xinyuan Tu, and Svetlana Yanushkevich. Dog identification using soft biometrics and neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[NZ08] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.