Tutorial for CNN Classification of Canine Ages Given Images of Teeth

Oasis E Husband, and Dr. Oge Marques

Florida Atlantic University, 777 Glades Rd, Boca Raton, FL 33431

Abstract

This paper aims to teach the programming of Convolutional Neural Networks using age classification in canine dental images. It utilizes the DogTeethAge dataset, a collection of 44 dental images from ages 0 to 18. This paper is structured with a step-by-step approach. The CNN achieves an 88.89% success rate in age classification, demonstrating its capability to predict canine ages accurately. This shows significant potential in using CNN to assist veterinary workers.

Associated Colab Project: Tutorial for CNN Classification of Canine Ages Given Images of Teeth

Vocabulary

- Image: A 3-dimensional matrix of values. Width, height, and color.
- Image processing: The structure of processing an image. This can be done in many different ways. Changing the color of an image is a kind of image processing.
- Image classification: Determining a visual fact about an image.
- Dataset: A set of data points or values. A list of information.
- Image augmentation: Changing around an image. Flipping an image is image augmentation.
- Seed: a random number that the layer will use. By inputting a seed, we can ensure our results will be consistent. This is easier for a walkthrough but is not technically necessary.

- Slice: One slice contains all the information for one instance of data. Each slice is like a plate of information.

- Transfer Learning: Taking the intelligence from one AI and using it as a base to build another.

- Epoch: A single iteration of training.

Step 1: Gather Data

Medical images that are of a quality high enough to be considered useful to an artificial intelligence are difficult to procure. Recently, a professional survey [1] has been performed, which states that many dog owners are uneducated about the act of pet cadaver donation. Additionally, a professional would need to take that cadaver, prepare it for study, and confirm that they are correct in the canines exact age. The entire process is time-consuming and difficult. This does not include the privacy concerns that follow medical information [2].

```
set(ages)
{0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18}
```

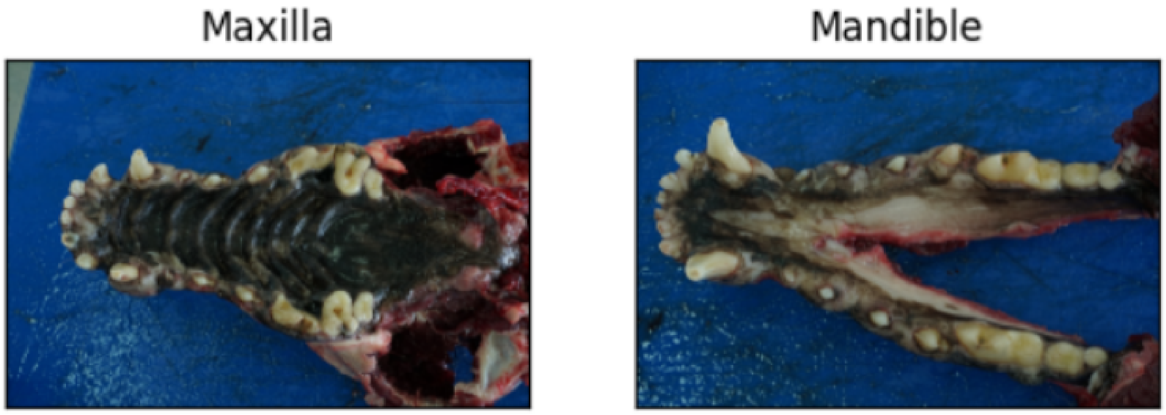
This tutorial uses the public DogTeethAge dataset from an open research project [3]. It provides 44 images, already randomized. Note that this dataset does not contain an even spread of ages.

There is no 6, 16, or 17, which means the program will be unable to learn how to guess these ages, as it will never be correct if it guesses one of these numbers. Additionally, there is only one tooth image for numbers 0 and 18, so it will be more difficult to decide to guess those numbers.

Step 2: Image Pre-Processing

Pre-processing is the most important step of any image classification algorithm. It is taking the images gathered in Step 1 and making them readable to the program.

Currently, a single dog is divided into two separate images of the maxilla and mandible.



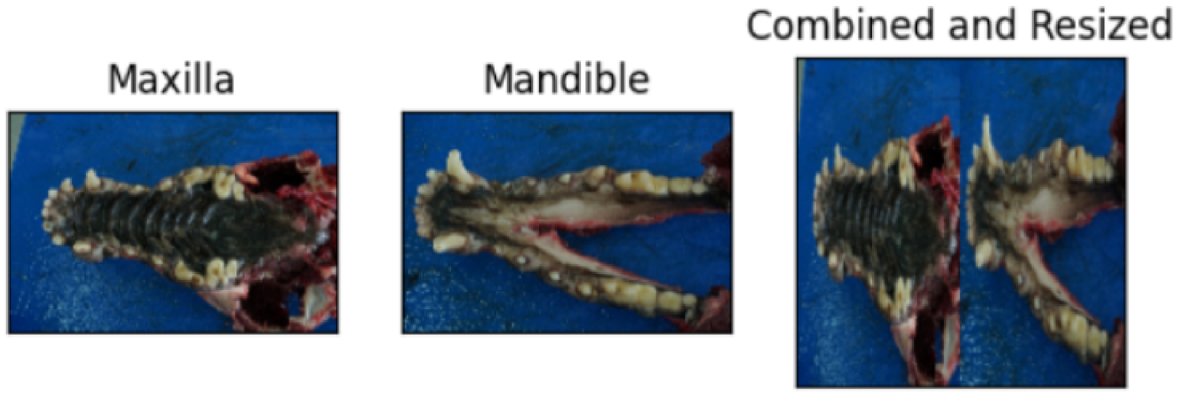
It is much simpler to combine these images together into one. Additionally, the current size is quite large.

```
maxillary.shape
(4000, 6000, 3)
```

The AI program will need to take in information from each pixel. However, many of them are unnecessary. We can combine both images together and reduce the size in one iteration through the data.

```
images = [] # Create an empty list

# Gets file names of current and next image and iterates through them.
for tooth_0File, tooth_1File in zip(image_names, image_names[1:]):

  tooth_0Path = os.path.join(image_path, tooth_0File) # Gets path to current image
  tooth_1Path = os.path.join(image_path, tooth_1File) # Gets path to next image

  # Checks if they're the from the same set
  if tooth_0File[0:4] == tooth_1File[0:4]:

    tooth_0 = cv2.imread(tooth_0Path) # Gets the pixel matrix from this path
    tooth_1 = cv2.imread(tooth_1Path) # Gets the pixel matrix from this path

    # Shoves images together into newImage
    newImage = combine_images(tooth_0, tooth_1)

    # Resizes newImage to (300x300x3)
    newImage = cv2.resize(newImage, dsize=image_size_2d)

    # Adds image to list
    images.append(newImage)
```

Now, although it is much smaller, every pixel holds valuable information.



Also, note that every image this program sees will be processed in this way. So, the variableness that might confuse a person (e.g., tall canines, short molars) is perfectly normal to the program.

Step 3: Database Pre-Processing

As hinted at in Step 1, right now, there are not very many images to work off of.

```
len(images)
44
```

Only 44. One person can not learn from only 44 different image examples, so neither can an artificial intelligence.

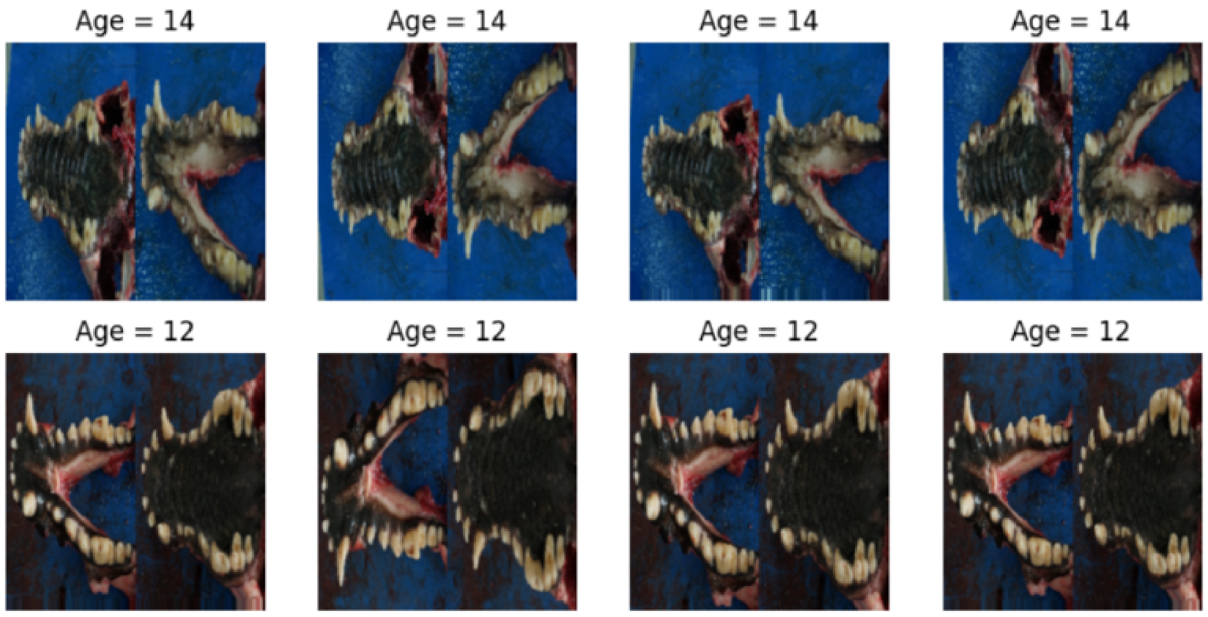
We can trick the program into having more information by copying the image multiple times and performing image augmentation, which essentially turns those copies into entirely new images. We will be doing image augmentation by using layers.

```
layers.Layer()
<keras.src.engine.base_layer.Layer at 0x785d4f02add0>
```

A learning algorithm is just a series of layers that the information runs through. Some layers are like a water filtration system, removing unnecessary or bad data. Some layers are scanners, learning something from the information. Some layers are shakers, moving and changing the information slightly. These are what we will implement now.

```
augmentationLayers = Sequential (
  [
    layers.RandomFlip("vertical", seed=3141), # Flips image
    layers.RandomContrast(0.2, seed=3141), # Lowers or brightens constrast
    layers.RandomRotation(0.001, seed=3141), # Rotates the image very slightly
    layers.RandomZoom(0.1, 0.01, 'nearest', seed=3141) # Zooms in slightly
  ]
)
```

Each layer randomly does something different to the image. The flipped ones are the most obvious, but try looking at the sides and corners and see if some are closer or further away than others.



Each of these images is wholly unique to an image processing algorithm.

Step 4: Combine and Split Data

The algorithm used later cannot take in the images and ages separately. The images and their ages will be formatted into singular slices of data. Each slice will contain both sets of teeth and, its age.

```
dataset = tf.data.Dataset.from_tensor_slices((images_augmented, ages_long))
```

Then, we split the program into a training set and a test set. Essentially, we are taking some of the dataset and putting it aside, so we can check if the program actually worked.

This will be random, so the program gets a full range of images. Although the ages were randomized in the first place, they were copied and changed four times, which can skew results. For example, if four sets of teeth of age 0 were in the training data, and none were in the testing data, then training the algorithm to detect when a canine is of age 0 has no use, and can also not be verified to work.

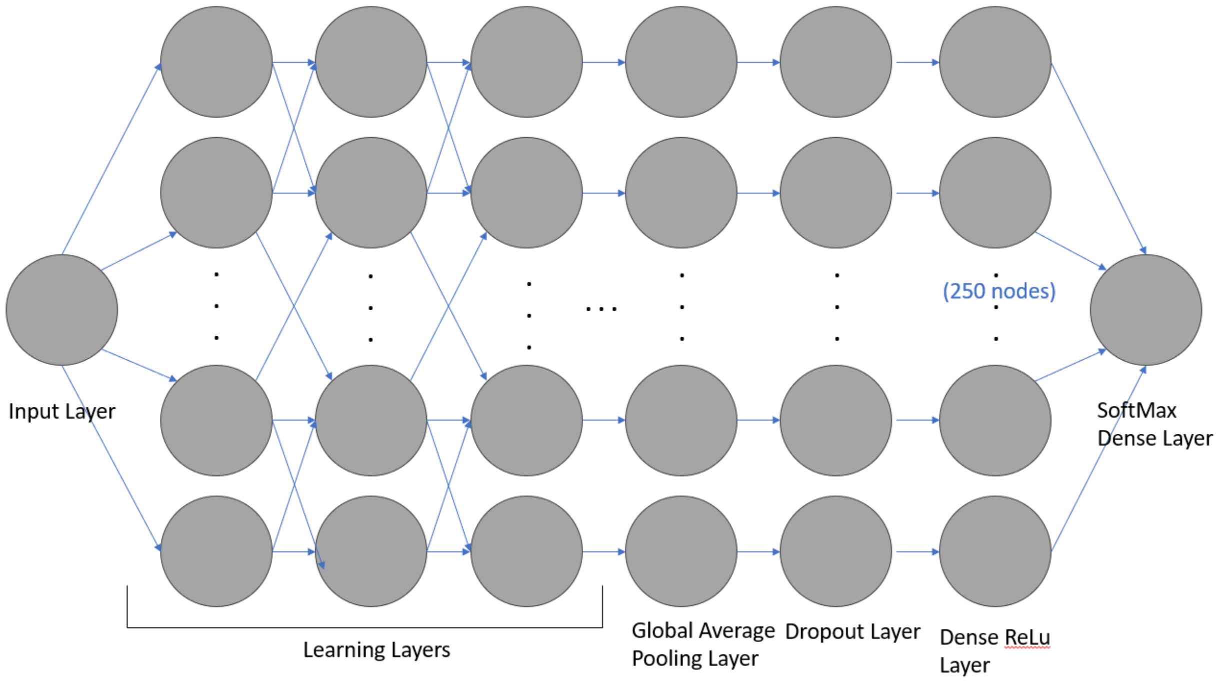
```
train, test = tf.keras.utils.split_dataset(
  dataset,
  left_size = 0.90, # Percentage of training set (90%)
  shuffle = True, # Randomizes the split
  seed = 3141
)
```

Now, instead of having 44 image pairs overall, the algorithm can use more than three times that while still having some test cases for afterward.

Crucially, there is even more processing that could have been done to make the dataset even more robust. If one patrols through the data by hand, one would notice that not all jaws are flipped in the same direction. Though it isnt necessary, as seen in Part 6, we could have flipped each image to quadruple the amount of data. A much bigger dataset would have taken much more time, as well, so it is not being done here.

Step 5: Build the Model

A neural network can be thought of as a collection of neurons that connect to each other in layers [4]. A Convolutional Neural Network (CNN) is similar. However, it is better suited to observing more complex sets of data, such as images, because Convolutional Layers have programs that can detect features from individual pixels [5]. We have seen layers before, when augmenting the image. However, these were single-node layers, similar to the last few in the given diagram.



Every circle is a node, and every column is a layer. Each node carries information on which node to connect to next, called weights. Transfer learning means we are taking these nodes and weights, which have already learned how to take in data effectively, and changing the last few layers to fit our own algorithm. There are many layers, but because we are using transfer learning, we only need to know about a few.

- InputLayers whole job is to take the database in. It is expecting size (224, 224, 3), but will work with any size.
- GlobalAveragePooling2D takes every value from an image and dumps it all into a pool, which is a single value. This keeps the program from overanalyzing unnecessary data and helps it focus less on the exact location of features. The algorithmic version of soft eyes.
- Dropout. When we run the dataset through this model, we will actually do so multiple times, or multiple epochs. To keep the program from relying on one piece of information too much, we will discard random slices.
- Dense layers are just fully connected to the previous layer.
- ReLu stands for Rectified Linear Unit, which essentially stops the program from continuing down a path where it makes an incorrect guess.

- SoftMax translates output data into probabilities. This is the last layer, so it needs to be in somewhat of a readable format.
- From GlobalAveragePooling2D and on, we will remove their layers and build our own. Essentially, we want to use the intelligence that this model already has, and use it to determine new information. Hence, transfer learning.

```
base_model = MobileNet(
    weights = 'imagenet',
    include_top = False, # Automatically throws away the last few layers
    input_shape = image_size_3d # Allows a different size for an input.
    #In this case: (300, 300, 3).
    )
```

ImageNet is a giant image database that many pre-trained models trained with.

We set the layers in the base_model to not be trainable. This is because it would be inefficient to re-train it. It has already learned all it can. We are just using that learning to help guide the new layers.

The constructed model corresponds to the diagram above.

```
model = Sequential(
    [
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dropout(0.5), # Dropping out 50% of the dataset per epoch
        layers.Dense(256, activation='relu'), # 256 determines how complex
        #the model is (how many different parts of the image is it focusing on?)

        layers.Dense(ages_max, activation='softmax')
    ]
)
```

Compiling the model prepares it to receive data.

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

- Adam is just the name of a commonly used optimization algorithm.

- Sparse_categorical_crossentropy simply means that our data is in number form (1, 2, 3...) instead of ([1 0 0], [0 1 0], [0 0 1]).

Now, to fit the data. The program runs the dataset through the model multiple times (epochs).

- Batch just runs the data through in chunks to make it go faster and use up less memory.

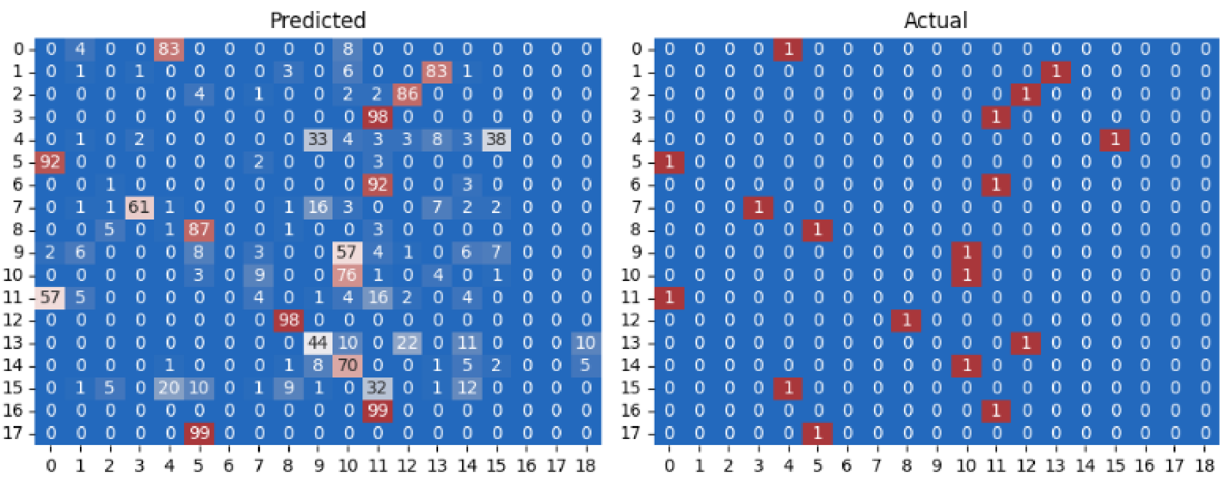
- Verbose means it will tell us more information.

```
model.fit(train.batch(8), epochs=20, verbose=2)
```

Step 6: Determine Results

To test our model, we use the test dataset sectioned off earlier, and run it through the models prediction function.

```
predictions = model.predict(test.batch(8))
3/3 [==============================] - 2s 330ms/step
```



As we can see, the algorithm performed fairly well. In some places, such as tests 0-4, the algorithm was fully confident. In others, especially 13 and 15, it was less so, which comes to an 88.89 success rate. Also notice, as mentioned in Step 1, the program never guesses 2, 16, or 17, because there are no references to go off of.

Conclusion

In conclusion, this educational paper has provided a comprehensive overview of the process involved in creating and implementing a Convolutional Neural Network (CNN) for the purpose of age classification in dental images. The goal was to provide a step-by-step introduction to the coding aspect of image processing to someone in the veterinary field.

References

[1] Tracey, K, Aktan, Í. Dog owners awareness of and motivations towards pet cadaver donation. Vet Rec. 2023;e3267. https://doi.org/10.1002/vetr.3267

[2] N.M. Safdar, J.D. Banja, C.C. Meltzer. Ethical considerations in artificial intelligence. Eur J Radio, 122 (2020), 10.1016/j.ejrad.2019.108768. 20192021

[3] Department-Of-Vet-Pathology-Unizg. Department-of-Vet-Pathology-Unizg/Dog$_{a}ge : DogsTeethAgeEst$
$of - vet - pathology - unizg/dog_{a}ge.Accessed1Dec.$2023.

[4] Abdi, H. (1994). A neural network primer. Journal of Biological Systems, 2(03), 247-281.

[5] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.