UFCFSN-15-3 Artificial Intelligence for Creative Technologies

A CNN classification on multiple species of frogs

Kieran Bowsher 19036291

Introduction

Description and intentions

The idea behind the project is to have a trained neural network that can classify different species of frogs from user submitted photos. This AI would be used inside a website or an application where users can take photos of frogs to find out what species it is. As well, this AI can also have other uses for biologists and wildlife experts to help them quickly identify each species of frog in the wild.
The AI could work alongside online databases of frogs to retrieve information to the user about their:

- habitats
- diet
- whether it is poisonous or venomous
- whether or not it can be held
- and if that species is endangered.

Project aims

The aim of this project is to create an AI classifier using a Convolutional Neural Networks (CNN) and using the findings from this report's research to help with the implementation. The AI will also be trained and tested on a dataset of frogs with a test score to see how well the AI did. The dataset will be handmade using photos of different frog species found online using search engines like Google.

Research

Previous work

Convolutional neural networks have been used a lot in image classification. There have already been some examples like (El et al, 2019) which uses Deep Convolutional Neural Networks (DCNNs) to classify 3 classes of animals: fish, reptile and amphibian. When it comes to identifying animals in the wild like amphibians or birds. The sounds they make are used to help identify them. There have already been lots of research and implementations done on animal call sounds like (LeBien et al, 2020), (Chao et al, 2019), (Huang et al, 2009) and (Xie et al, 2019), all of which map the sounds the animal makes to a spectrogram and use CNNs to classify the sound based on the wavelengths and frequency.

However, there is not much research done on identifying animals like frogs based on their appearance. While there are some examples of CNN for big cats (Sovit Ranjan Rath, 2019), the only other examples for frogs are for their call sounds like (LeBien et al, 2020) and (Chao et al, 2019). Research was done by (Cannavò et al, 2012) to look into identifying different frogs using the textures and patterns of their skin. The findings were to have all images of frogs to be black and white and extract a certain part of the image that contained the frog's skin. This extracted image could then be used to identify the frog.

Methods

Dataset

The current dataset used for training the network is handmade from a selection of images from Google and has around 517 images. The dataset currently has 4 classes of frog species which include: Desert Rain Frog, Red Eyed Tree Frog, Glass Frog and Tomato Frog and more classes can be added to the dataset for training. All the images in each class are separated using a 70/15/15 ratio for three folders called Training, Validation and testing.

Data Augmentation

All the images are altered when loaded for training using either Pytorch's Colour jitter or Grayscale transform functions. By using these transforms, it prevents the network from learning any biases from the images like the colours of the frogs.

Convolutional Neural Network

In order to train a CNN with satisfying results, the Python based library Pytorch is used greatly to help with implementing the network. Along with this, to accelerate the rate at which the network trains, a GPU is used. Currently Google Colab uses a Nvidia Tesla K80 GPU but if this implementation is used on local hardware, then potentially a better GPU can be used which will increase training rates.

SmoothGradCAMPlusPlus

In order to help with understanding how the network is identifying the images, a Python based GitHub project called SmoothGradCAMPlusPlus is used. This project can load an image and the network model and produce heat map images showcasing where the network looks when identifying images as shown in **Figure 05**.

Transfer Learning

An important part of this project is the usage of transfer learning (Pranshu, S 2021). The idea behind transfer learning is to use previous trained models on large datasets and use it to solve a different but related problem and using transfer learning also help speed up the training times

Implementation

The convolutional neural network is implemented using Google Colab and uses the Python programming language. Both the training, testing and validating of the network and SmoothGradCAMPluPlus are implemented inside the same Google Colab file. As well, the training, testing and validating of the network is also implemented using Spyder and Pytorch on local hardware which uses a Nvidia GTX 1060 6GB GPU. Allowing the network to be implemented on local hardware will also help with performance comparison later in this report.

Training, Validation and testing of the network

```
train_transform = transforms.Compose([transforms.Resize(image_resize),
                                      transforms.RandomRotation(5),
                                      transforms.CenterCrop(image_crop),
                                      transforms.Grayscale(num_output_channels = 3),
                                      transforms.RandomHorizontalFlip(),
                                      transforms.ToTensor(),
                                      transforms.Normalize(mean_normalize, std_normalize)])
```

Figure 01. Transform used when loading the dataset

Before the network is trained, the dataset is loaded from Google Drive or the folder on local hardware. One of the most important steps in training the network is describing how the images will be transformed. **Figure 01** showcases this with different types of transforms like RandomRotation, RandomHorizontalFlip, Grayscale, etc. There is an issue with the usage of CenterCrop since it sometimes crops out very important parts of the image.

The grayscale transform is very important since it turns all the images used for training into black and white images. This was done to prevent the network from learning the colours of each frog and forces the network to learn the patterns of the frog's skin similar to (Cannavò et al, 2012). An example is shown with **Figure 02a** and **Figure 02b** as two of the Desert Rain Frog images have different patterns on their skin. It can be argued that using the ColorJitter transform can also be used since it randomly changes the colour of each frog. The Results and findings section of this report will showcase the finding between Grayscale and Colour Jitter transforms.

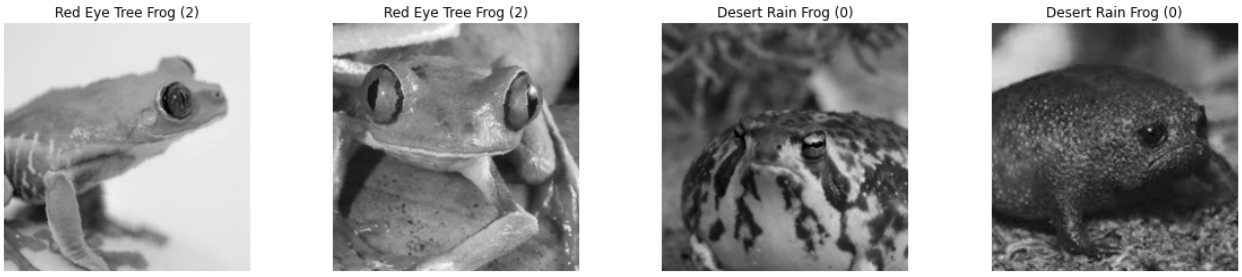

Figure 02a. Example of transformed training dataset

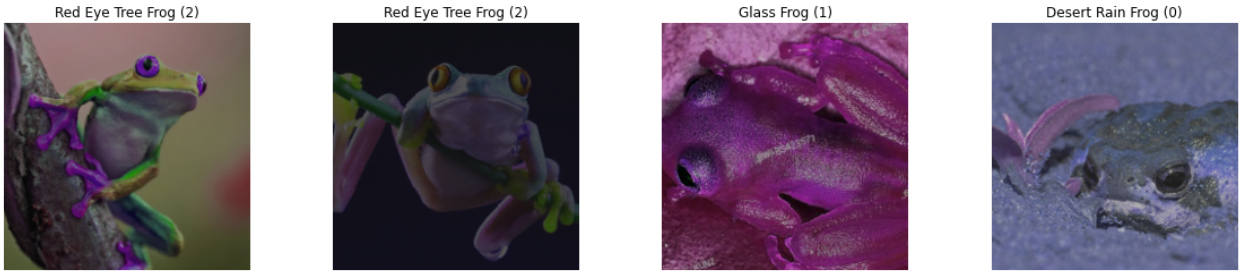

Figure 02b. Example of transformed training dataset

The network is trained using a pre-trained convolutional neural network called ResNet18 which has 18 layers. ResNet18 has been trained on more than a million images using the ImageNet database and allows the network to classify 1000 different objects. Using ResNet18 also means that the network will be quicker at learning the images. For training, the network is trained for 100 epochs with a batch size of 8 and a learning rate of 0.001. Once training is completed a graph is produced to showcase the training and validation loss as shown by **Figure 03.**

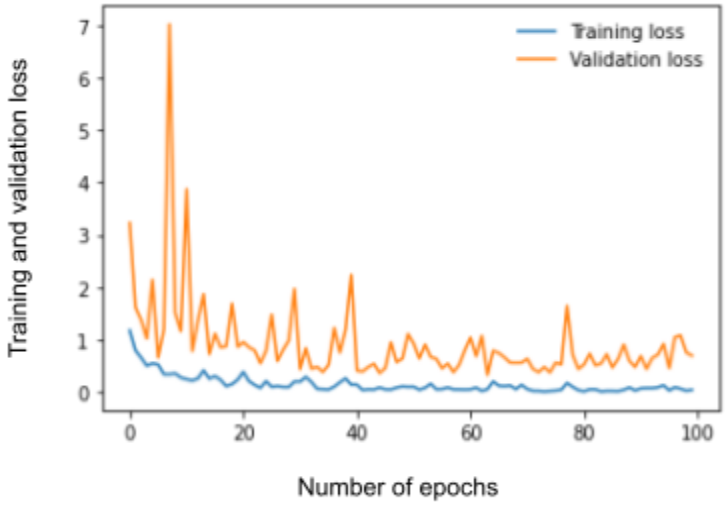

Figure 03. Example of a graph for training and validation loss during training

Once training is complete, the next part is testing the trained network model. Testing is done using the images from the testing folder of each class and each test is given a percentage as shown in **Figure 04**. This will help with getting a general idea on how well the network model has done.

Class	Number of incorrect image classification	Total images used for testing	Test percentage
Desert Rain Frog	18	19	94%
Glass Frog	15	16	93%
Red Eyed Tree Frog	19	20	95%
Tomato Frog	10	15	66%
Overall test percentage	62	70	88%

Figure 04. Example of test accuracy

SmoothGradCAMPlusPlus

Finally, SmoothGradCAMPlusPlus is used to help identify where the trained model is looking when trying to identify an image. **Figure 05** already shows an example of this and it can be seen that the trained model is looking at Frogs skin texture on the left of the image. With refinements and longer training, the results from SmoothGradCAMPlusPlus should improve.

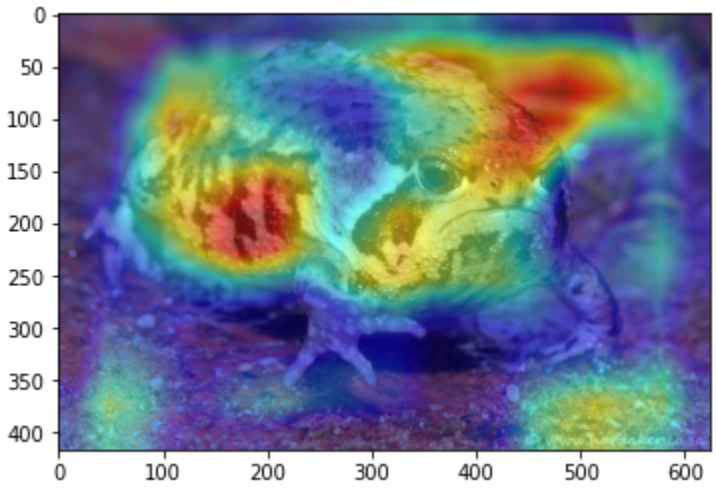

Figure 05. Example of SmoothGradCAMPlusPlus

Results and findings

Training time performance

As already discussed in this report, the network was trained using Google Colab hardware and local hardware. The parameters used for training on both systems were an epoch of 100, learning rate of 0.001 and a batch size of 8.

Image transform	Google Colab	Local Hardware
Grayscale	19 Minutes	13 Minutes
Colour Jitter	23 Minutes	15 Minutes

Figure 06. Comparison of training performance

As shown by **Figure 06** training the model using local hardware produces faster training time. This is mainly due to the fact that the Nvidia GTX 1060 6GB has faster hardware than a Nvidia Tesla K80 12GB but the Tesla K80 12GB has twice as much VRAM so its usage in higher resolution CNN training is better.

Grayscale and colour jitter transforms comparison

The tests involved training and testing the network multiple times using two transforms called Colour Jitter and Grayscale. The purpose of the test is to see if there are any differences and improvements when using one transform over another. Each result of the test was recorded using the produced loss graph as shown by **Figure 07** and a text output of a resulting percentage as shown by **Figure 08**. In this report, only the best results from the tests are shown.

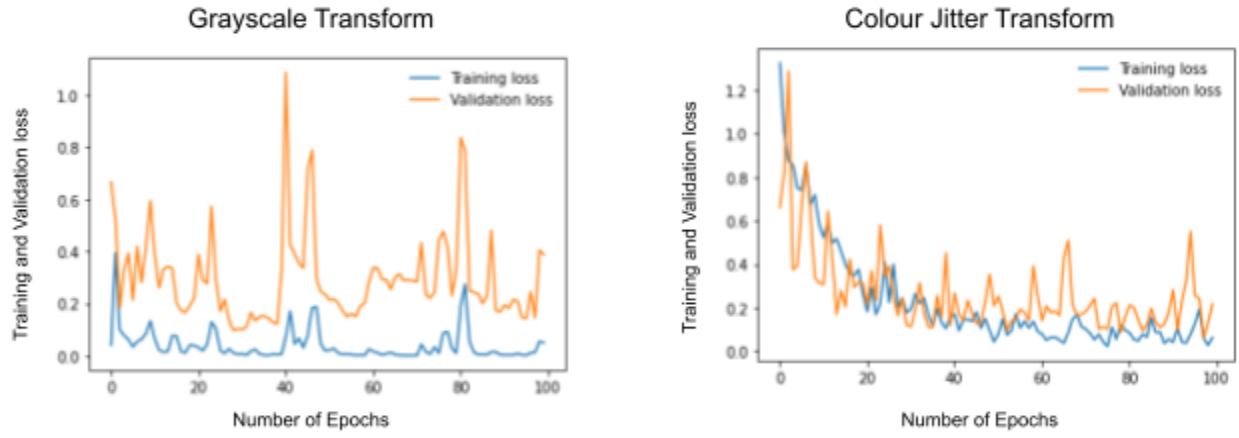

Figure 07. Comparison between Grayscale and Colour Jitter training and validation loss

Figure 07 showcases two graphs that show the loss during training and validation for the two models that were trained on grayscale and colour jitter transforms. Clearly the model using colour jitter had a nicer slope of loss decrease while the model using grayscale generally stays flat with lots of spikes of loss during training.

Image transform	Desert rain frog	Glass Frog	Red Eyed Tree Frog	Tomato Frog	Overall
Grayscale	90% (20/22)	76% (19/25)	88% (32/36)	93% (27/29)	87% (98/112)
Colour Jitter	86% (19/22)	88% (22/25)	97% (35/36)	93% (27/29)	91% (103/112)

Figure 08. Comparison between Grayscale and Colour Jitter test accuracy

As shown in **Figure 08**, the model that uses the colour jitter transform performs better in testing than the model that uses the grayscale transform. Although this is true for the overall accuracy, when looking at the individual test with each class there are some differences. One of these differences is that the model using grayscale transform is better at classifying Desert rain frogs while the model using colour jitter is better at classifying Glass frogs and Red Eye Green frogs.

GradCAMPlusPlus findings

Below are some sample images that were used to test the two models during testing which are being used by SmoothGradCAMPlusPlus to show where the models are looking when classifying each image.

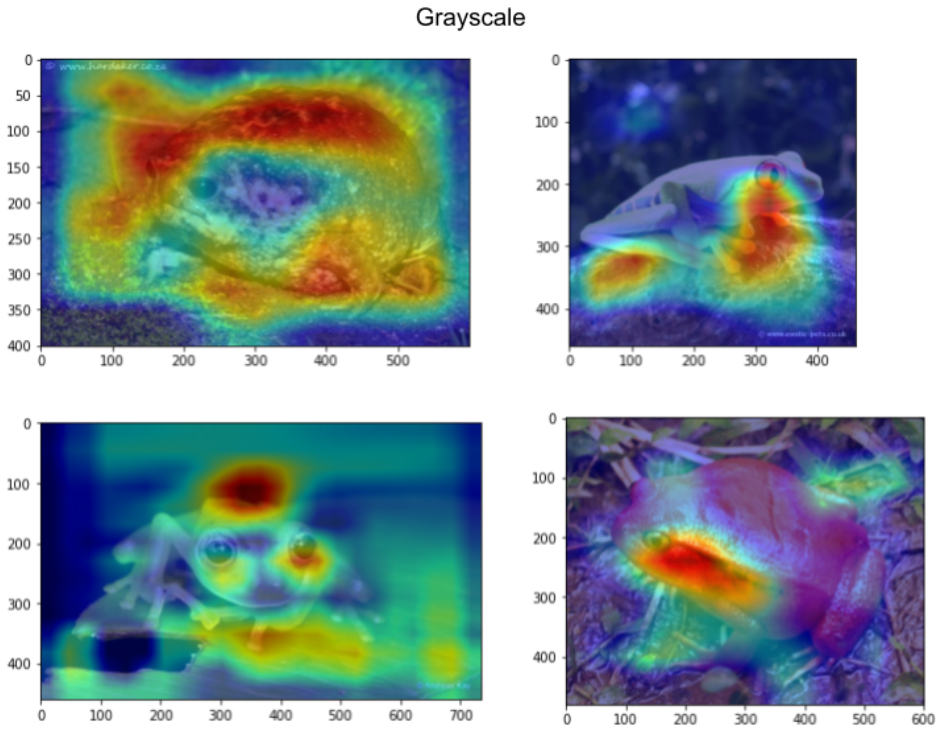

Figure 09. Heat maps produced from SmoothGradCAMPlusPlus. In order from top to bottom, Desert rain frog, Red eyed tree frog, glass frog and tomato frog

As shown by **Figure 09** which is produced from the grayscale model. The findings for both the Red eyed tree frog and Glass frog are mixed with the model looking at different parts of the image. This finding can be reinforced by **Figure 08** as the test done for these two frogs are the lowest out of the four tests.

However, the findings for the desert rain frog are very good as it looks exactly at the skin patterns on the frog. This can be somewhat true for the Tomato frog as it looks at the skin pattern near the frog's face.

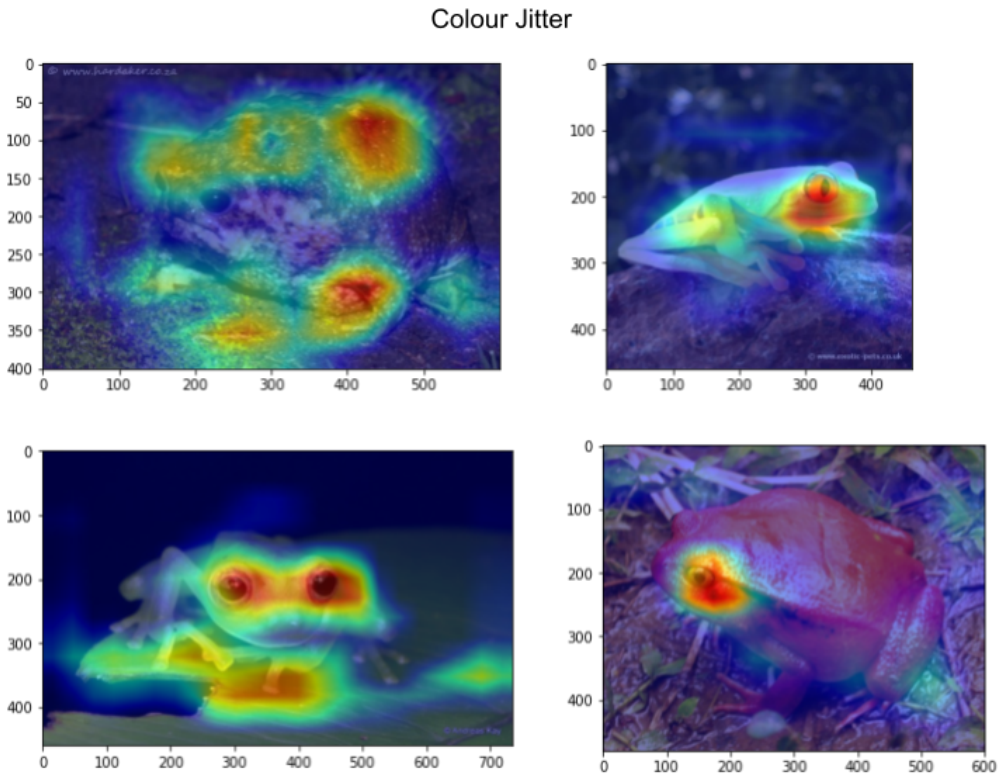

Figure 10. Heat maps produced from SmoothGradCAMPlusPlus. In order from top to bottom, Desert rain frog, Red eyed tree frog, glass frog and tomato frog

Figure 10 which is produced from the colour jitter model shows that this model will generally identify the frogs using their eyes. This is not true for the Desert rain frog with the model looking at the skin pattern. This can be somewhat reinforced when looking at **Figure 08**, since it shows in those tests that the Desert rain frog had the lowest percentage.

Conclusion

The project has been a great learning experience into neural networks and convolutional neural networks. **Figure 11** showcases the visual results and the findings explained in the report overall show very good results. There can be a lot of improvements made to this like using another pre-trained model for example GoogLeNet. Since this project mainly used pre-trained models, it could be argued that using a custom made CNN and training it for a long time will yield greater results.

As well, the dataset could be improved a little bit with adding more images and classes but this can easily be expanded later on. Overall, the model produced from this project meets the general requirements asked from the report.

Figure 11. Visual results from testing

References

Nidhal Khdhair El abbadi & Elham Mohammed Thabit A. Alsaadi (2019). Auto Animal Detection and Classification among (Fish, Reptiles and Amphibians Categories) Using Deep Learning. [Online]. Available from:
https://www.researchgate.net/publication/337052841_Auto_Animal_Detection_and_Classification_among_Fish_Reptiles_and_Amphibians_Categories_Using_Deep_Learning
[Accessed 17 December 2021].

Jack LeBien, Ming Zhong, Marconi Campos-Cerqueira, Julian P. Velev, Rahul Dodhia, Juan Lavista Ferres, T. Mitchell Aide (2020). A pipeline for identification of bird and frog species in tropical soundscape recordings using a convolutional neural network. [Online].
Available from: https://www.sciencedirect.com/science/article/pii/S1574954120300637
[Accessed 17 December 2021].

Kuo-Wei Chao, Yi-Chu Chao, Chin-Kai Su, Nian-Ze Hu, Wei-Hang Chiu (2019). Using Machine Learning Method to Identify for Frog Classification. [Online].
Available from https://ieeexplore.ieee.org/document/8942750
[Accessed 16 December 2021]

Chenn-Jung Huang & Yang, Yi-Ju & Yang, Dian-Xiu & Chen, You-Jia. (2009). Frog classification using machine learning techniques. [Online]. Available from:
https://www.researchgate.net/publication/222925491_Frog_classification_using_machine_learning_techniques
[Accessed 16 December 2021]

Flavio Cannavò, Giuseppe Nunnari, Izzet Kale, F. Boray Tek (2012). Texture Recognition for Frog Identification. [Online]. Available from:
https://projet.liris.cnrs.fr/imagine/pub/proceedings/ACM-MULTIMEDIA-2012/maed/p25.pdf
[Accessed 16 December 2021]

Xie, Jie & Towsey, Michael & Zhang, Jinglan & Dong, Xueyan & Roe, Paul. (2015). Application of image processing techniques for frog call classification. [Online]. Available from:
https://www.researchgate.net/publication/293491557_Application_of_image_processing_techniques_for_frog_call_classification
[Accessed 16 December 2021]

Yann LeCun, Corinna Cortes, Christopher J.C. Burges. THE MNIST DATABASE of handwritten digits. [Online]. Available from:
http://yann.lecun.com/exdb/mnist/
[Accessed 22 December 2021]

Sovit Ranjan Rath, (2019). Wild Cats Image Classification using Deep Learning. [Online]. Available from: https://debuggercafe.com/wild-cats-image-classification-using-deep-learning/
[Accessed 22 December 2021]

Pranshu, S (2021). Understanding Transfer Learning for Deep Learning. [Online]. Available from: https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/
[Accessed 22 December 2021]

Bibliography

TensorFlow. TensorFlow Datasets: a collection of ready-to-use datasets. [Online]. Available from: https://www.tensorflow.org/datasets/catalog/overview
[Accessed 22 December 2021]

Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, Kaori Togashi (2018). Convolutional neural networks: an overview and application in radiology. [Online]. Available from: https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9
[Accessed 22 December 2021]

Pantechsolutions (2021). Animal Detection and Classification using Deep Learning. [Online]. Available from:
https://www.youtube.com/watch?v=K6cGZgEch-4
[Accessed 22 December 2021]

Miao, Z., Gaynor, K.M., Wang, J. et al. (2019). Insights and approaches using deep learning to classify wildlife. [Online]. Available from:
https://doi.org/10.1038/s41598-019-44565-w
[Accessed 28 December 2021]

Sumit Saha (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. [Online]. Available from:
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
[Accessed 28 December 2021]