Classifying Different Types of Mushrooms Using CNN

Adithya Murali
a9murali@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Abstract

Convolution Neural Network are the state of the art models when it comes to image classification. This paper aims to classify different species of Mushrooms using a CNN. An accurate prediction is of utmost importance as there are many poisonous species of mushrooms. A human expert needs to spend a considerable time to learn about the different species mushrooms to classify them accurately. This project attempts to see just how well this task can be automated using a CNN on our dataset of 10 different species of mushrooms.

We compare the different CNN models for our task and find the best one while trying to keep the complexity of the model low. We find that Mobilenetv2 model performs the best when compared against other more complex models like VGG16, Resnet50, Xception and Inception-ResNet-V2. Mobilenetv2 gives an accuracy of 0.76 and the highest APM (Accuracy per million parameter) of 0.217 on our dataset.

Introduction

Mushrooms are the fruiting body of a fungi. Mushrooms are also consumed as food in many parts of the world. Mushroom hunting is a niche hobby where people forage for mushrooms for cultivation, display and consumption. However, care must be taken while handling mushrooms as many species of mushroom are poisonous and can be potentially fatal. Around 10,600 people were poisoned by mushrooms and 22 have died from 2010 to 2017 in France according to this study (Sandra Sinno-Tellier 2019).

Classifying mushrooms correctly is an important task which requires plenty of skill. Making a classification error can be a potentially deadly mistake. The new state of the art Convolution Neural Networks are an excellent choice for this classification task and they have the potential to match or even surpass the accuracy of a human expert.

This project makes an attempt to use transfer learning on various state of the art CNNs to classifying pictures of mushrooms. VGG-16 is one such CNN model with 16 layers. This model was the winner of the ImageNet Challenge 2014 and has about 138 million parameters(Simonyan and Zisserman 2015a). ResNet is another unique CNN architecture which uses "residual connections"(He et al. 2015a). The research question this project will address is: Which CNN architecture and optimization method/loss function will best classify a given dataset of mushroom images correctly?

The state of the art CNN models usually come with pretrained parameters which this project later fine tunes to fit the project's domain. This process is called transfer learning. This project will use accuracy, precision and recall and perhaps their combination (F-score) to assess the performance of a model. The dataset is collected by downloading images of different types of mushrooms from a web search engine. This process can be automated by a script. The expected outcome of this project is to build a reliable application that can classify different types of mushrooms with a reasonable degree of accuracy.

The key findings from this project was that we can indeed build a CNN model that can classify different species of mushrooms with a reasonable degree of accuracy. The second key finding is that the MobileNetV2 model can be used for this task which is very convenient if this model is run locally on a device with low computational power.

The key contributions made by this project are:-

1. Created a new dataset of 1000 mushroom images from 10 different species using images from Flickr.
2. Used a CNN trained on Imagenet data to classify mushrooms. Fine-tuned the CNN further to work better with our dataset.
3. Used a novel new metric called Accuracy Per Million Parameters (APM) to measure the trade-off made between performance and complexity.

Related Work

LeNet-5 is one of the simplest CNN architecture with just 5 layers (Lecun et al. 1998). This architecture consists of convolution layer followed by max-pool layer which nearly every modern state of the art CNN uses. There are traditional dense layers at the end to produce an output. This model was first published in 1998 and was used for character recognition.

VGG-16 is a 16 layer deep CNN with 13 convolutional/max-pool layers and 3 dense layers (Simonyan and Zisserman 2015b). The main advantage of this model is its depth compared to other models at that time.

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Inception-v1 is a 22 layer neural network with 5 million parameters which set a new record in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14)(Szegedy et al. 2014). This model increased the depth and width of the neural network without much increase in computational cost. This was done by approximating the expected optimal sparse structure. There are many other neural networks such as Inception-v5(Szegedy et al. 2015), ResNet-50(He et al. 2015b), Xception(Chollet 2017), Inception-ResNet-V2(Szegedy et al. 2016) which merges the two architectures and ResNeXt-50(Xie et al. 2017). They vary in their design and performance but have the same underlying design of feeding images through a series of convolution/max-pool layers and finally passing it through a series of dense layers to make a prediction.

It is also worthwhile to look at the various optimization techniques that is available to us to optimize the weights. Adam's optimization(Kingma and Ba 2017) is a more efficient variant of Stoichastic Gradient Descent. This optimization method combines AdaGrad(Duchi, Hazan, and Singer 2011) and RMSProp(Tijmen Tieleman)

This project will compare the various CNNs mentioned above for the task of classifying mushrooms and this project will use Adam's optimization to optimize the parameters of the neural network.

This paper (Preechasuk et al. 2019) managed to classify 45 different types of mushrooms with 0.78, 0.73 and 0.74 of precision, recall and F1 score, respectively. They constructed a custom model of around 7 layers and used data augmentation by flipping and rotating images.

Another paper (Ottom and Alawad 2019) used 3 different types of classifiers which neural networks, K-Nearest Neighbour and Decision tree to classify mushroom species. They found that KNN seems to give the best results.

Another such study (Dong and Zheng 2019) classifies Enoki Mushroom Caps using CNN and their study produced promising results.

Methodology

This project implements the **Mobilenet V2**(Sandler et al. 2018) CNN. The rationale behind choosing this model of CNN is its small size compared to other state of the art CNNs such as VGG-16(Simonyan and Zisserman 2015a). Mobilenet V2 has 3,538,984 parameters which is orders of magnitude smaller than the number of parameters in VGG-16(138,357,544) or ResNet-50(He et al. 2015b) with 25,636,712 parameters. Mobilenet V2 was developed with the idea that the CNN should be able to run with low computational power without sacrificing much performance which is ideal for our task. Mushroom foragers usually do not have access to much computing power and sometimes they do not have access to the internet when foraging in rural areas. Therefore, being able to run a CNN on a mobile device with low processing power is very preferable. We will compare the performance of this lightweight model against other heavyweight models like **VGG-16**(Simonyan and Zisserman 2015a),**ResNet-50**(He et al. 2015b),**Xception**(Chollet 2017) and **Inception-ResNet-V2**(Szegedy et al. 2016).

We have chosen 10 different classes of mushrooms for our classification task out of which 3 are deemed to be poisonous or fatal when consumed. The chosen species of mushrooms are

1. Amanita caesarea
2. Armillaria mellea
3. Boletus badius
4. Clavariaceae
5. Calvatia gigantea
6. Clavulinaceae
7. Fistulina hepatica
8. Amanita phalloides (Poisonous)
9. Cortinarius rubellus (Poisonous)
10. Amanita bisporigera (Poisonous)

The three poisonous mushrooms chosen are 3 different species which can be easily confused and classified as safe to consume by humans which is the rationale behind choosing them for our classification task.

The images of these pictures were acquired by scraping the popular image hosting website Flickr using a python script (Jocher 2020). A total of 100 images were retrieved for each mushroom species. There were some inaccuracies during this step where certain images did not match the given search query(species name). Such images were manually removed.

10 images from each class were set aside as test set. Out of the remaining images (90 if no images were manually removed), 80% were used as training images and 20% of the images were used for validation.

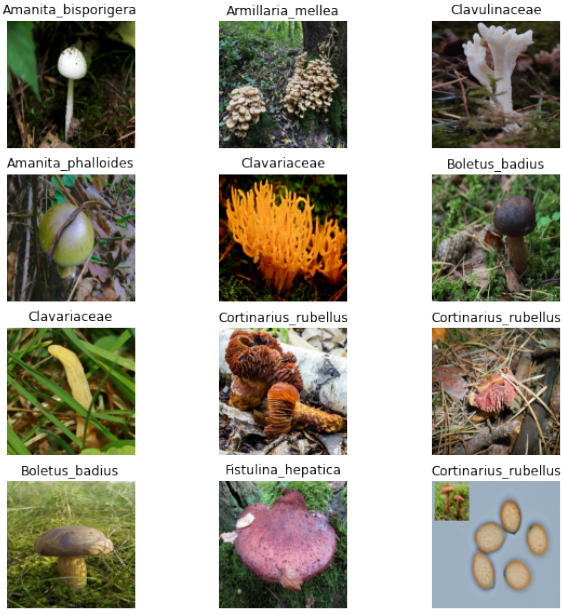

Figure 1: Sample of the images scraped from Flickr

The images are first sent through a layer which resizes them to the size 224×224×3 which is the default input image size for the Mobilenet v2 model. The three numbers stand for height,width and the number of colour channels.

The images are then sent through a an image augmentation layer where they are randomly rotated by a small degree. This data augmentation layers is used to make the model more robust and less prone to over-fitting. We do not perform data augmentation using other methods such as adjusting brightness or saturation. We perform data augmentation only for the training images during the training phase.

The image is then normalized to a value between -1 to 1 and fed into the Mobilenet V2 model. We do not use the "top" layer of the model as our context is different from its original design use. We need to classify the picture into one of the 10 different classes of mushroom whereas the model was originally used to classify 1000 different classes from the Imagenet dataset. It is important to note that the Mobilenetv2 model's weights are not updated during training so that we do not destroy the pre-existing weights. We only update the weights of our custom layers and evaluate it on the test dataset. Once we get satisfactory results on the test dataset, we unfreeze the Mobilenetv2 layer and train the model again with a very low learning rate. This is second phase of training is called fine tuning and may improve our model. This is the layer that will be replaced with other models when we test other models against Mobilenetv2.

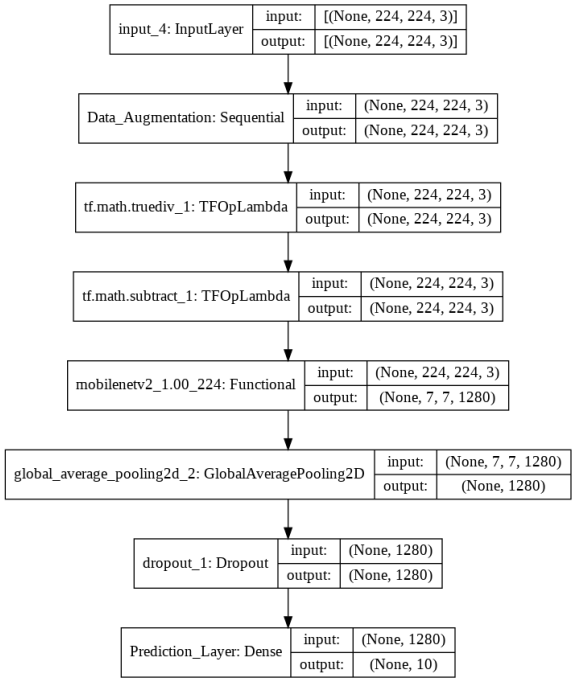

Figure 2: CNN architecture

The crux of the CNN model is in the mobilenetv2_1.00_224 layer in Figure 2. This is the Mobilenetv2 layer which takes in the preprocessed input image of size $224 \times 224 \times 3$ and given an output of size $7 \times 7 \times 1080$

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 3: mobilenetv2_1.00_224 layer architecture. Here n refers to the number of the times the given layer is repeated. c refers to the output channel size (third dimension) and s refers to the stride distance while applying convolution and t is an hyperparameter for that particular layer. This table has been adapted from (Sandler et al. 2018)

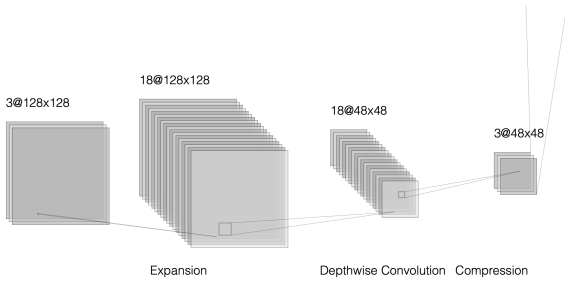

Figure 4: Bottleneck layer. We expand the channel, perform depthwise convolution and compress the channel and send it to the next layer.

The biggest advantage of this model is that it uses bottleneck layer which is also referred to as the inverted residual layer which is computationally cheaper to use compared to the standard convolution layer.

To convert a block of $M \times N \times k$ to a block of size $M' \times N' \times k'$ using standard convolution technique, we apply filters of depth k a total of k' times. We perform element wise multiplication between the filter and input block which can be computationally expensive.

Depthwise separable convolution splits this convolution step into two parts which gives approximately the same result as regular convolution at a much lower cost. We perform 2D convolution for each channel to convert the input block of size $M \times N \times k$ to a block of size $M' \times N' \times k$. We then perform regular 1×1 convolution across the depth to convert $M' \times N' \times k$ block to $M' \times N' \times k'$ which is also the final output of regular convolution. This was already implemented in Mobilenetv1.

Mobilenetv2 improves on this idea by performing a 1×1 regular convolution to expand the number of channels before passing it to the depthwise convolution layer. The output from the depthwise convolution layer is compressed by another 1×1 regular convolution layer and it is added to the original input if they have the same channel size. This compression limits the amount of information flowing across one layer to another which is the reason behind the name bottleneck layer.

The output from the model is then fed into a Global 2D Average Pooling layer which reduces the dimension of the output tensor by averaging it across the length and width for each sample and channel. We then send it through a dropout layer which randomly drops the activation signal of a certain number of nodes with a probability of 0.2. This is done to combat over-fitting of the data.

We then finally add a dense layer at the end with the same number of nodes as the number of classes. We can apply a softmax layer at the end to convert the real valued output to a probability distribution but we refrain from doing so as a softmax layer can be applied at the time of the estimation of the loss function. The softmax layer is defined as

$$\sigma(\mathbf{z})_i = \frac{e^{\mathbf{z}_i}}{\sum_{i=1}^{k} e^{\mathbf{z}_k}} \tag{1}$$

where k is the number of classes which is 10 in our case.

We use the categorical cross entropy loss measure which is defined as

$$L(\mathbf{y}, \mathbf{t}) = -\sum_{i=1}^{n} t_i log(y_i) \tag{2}$$

where $\mathbf{y}$ is the probability distribution of one sample across all the classes and $\mathbf{t}$ is the one hot encoding of the ground truth. We sum up this loss across all the samples to find the total loss.

We use Stoicastic Gradient Descent(SGD) with a learning rate of 0.01 to minimize our loss function. We use a learning rate of 0.001 during the fine tuning step. SGD is a well known algorithm in optimization which is better than regular gradient descent in its performance. SGD calculates the gradient at a sample point and updates the parameter based on the gradient at this point. This is computationally cheaper than calculating the gradient for the entire dataset for each iteration.

Results

We use the accuracy metric to judge the performance of our model.

$$Accuracy = \frac{Number\_of\_correct\_classifications}{Total\_number\_of\_samples} \tag{3}$$

We would like our model to generalize well, that is we want our model to classify each distinct specices correctly instead of a binary classification such as edible/non-edible which is the reason behind the usage of accuracy metric.

We also consider a simple new metric which is accuracy per million parameters. Since more parameters contribute to more memory usage and complexity, we would like to find out the contribution made by the additional parameters. This metric will be helpful to compare Mobilenetv2 against the more complex models.

$$APM = \frac{Accuracy \times 10^6}{Number\_of\_Parameters} \tag{4}$$

We have decided to train the model for a total of 20 epochs

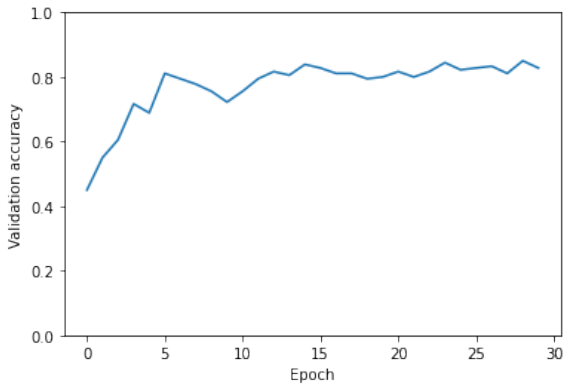

Figure 5: Epoch vs Validation accuracy

as there is no significant improvement in validation accuracy after that on our dataset. The learning rate of SGD for the transfer learning and fine tuning step has also been experimentally fixed to 0.01 and 0.001 respectively. There seems to no one optimization algorithm that performs the best for the particular optimization task in hand. Therefore, we chose to optimize with the simplest optimizer available which is SGD. We trained our neural network on the train dataset of 900 images of 9 different species of mushroom.

The below table shows the accuracy of different models on our test dataset of 100 images with 10 images from each class. All the models have the same architecture as in Figure 2 except for the difference in the 5^{th} layer. The details of the implementation have been described in the previous section.

Model	Accuracy
MobileNetV2	**0.64**
VGG16	0.63
Resnet50	0.70
Xception	0.61
Inception-ResNet-V2	0.65

Table 1: Accuracy of the models on our dataset

Fine tuning the models by changing the weights in the 5^{th} layer very slightly to work better with our context seems to give better results. The original models were trained on a different dataset. This leads to small increase in training time.

Model	Accuracy
MobileNetV2	**0.76**
VGG16	0.72
Resnet50	0.73
Xception	0.74
Inception-ResNet-V2	0.76

Table 2: Accuracy after fine tuning

The table below summarizes the performance of the models relative to its complexity. The parameters here refer to the convolutions filter weights and the associated biases in each layer of the CNN.

Model	Accuracy	Parameters	APM
MobileNetV2	**0.76**	**3.5M**	**0.217**
VGG16	0.72	138.3M	0.005
Resnet50	0.73	25.6M	0.028
Xception	0.74	22.9M	0.032
Inception-ResNet-V2	0.76	55.8M	0.013

Table 3: Accuracy per million metric

The table below shows the time taken to process 100 images by the different models. Google's Colab environment was used with Tesla T4 GPU acceleration to benchmark the performance of our different models.

Model	Time
MobileNetV2	**1.256 s**
VGG16	8.413 s
Resnet50	1.764 s
Xception	2.407 s
Inception-ResNet-V2	2.447 s

Table 4: Time taken to process 100 images.

The below figures explain the rationale behind our choice of learning rate and optimizer. A learning rate of 0.001 is too low whereas a learning rate of 0.1 is too high. The sweet spot appears to be around 0.01

We compared various optimizers and they all seem to perform the same and do not significantly affect our training process.

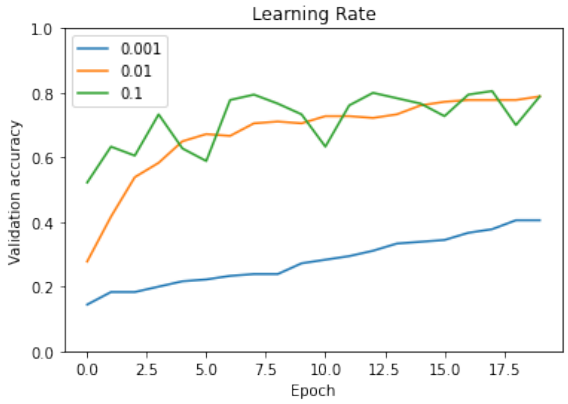

Figure 6: Learning rate

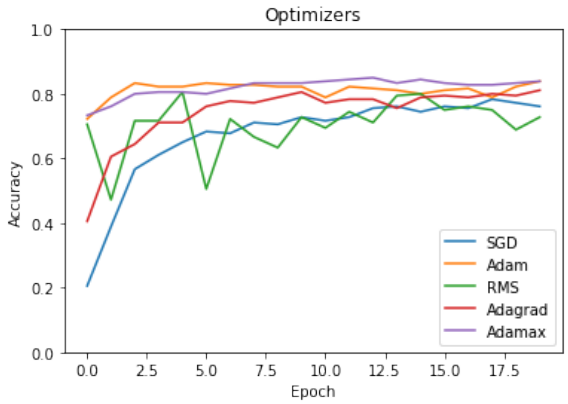

Figure 7: Optimization

Figure 8 shows where our model (MobileNetV2) did not work on our dataset. Red font denotes a poisonous mushroom misclassified as a safe species. We ran into just one instance of such misclassification out of the 30 images of poisonous mushrooms from our test dataset.

Other misclassifications appears to be between mushroom species which are very similar in their appearance.

Discussion

Our experiments suggests that a CNN can be used to classify different species of mushrooms with a reasonable degree of accuracy. Our dataset had 10 different classes and most of the misclassifications are between very closely related species which look like they can be misclassified by human experts as well. One way to overcome this problem is to take pictures from multiple angles if we have physical access to the mushroom.

Secondly, such a model can be used to augment human knowledge rather than completely replace it. It is worthwhile to note that the CNN does not take into consideration the smell, texture and the taste of the mushroom which an human expert has access to. Feeding in such information can improve the performance of our model. Another way to im-

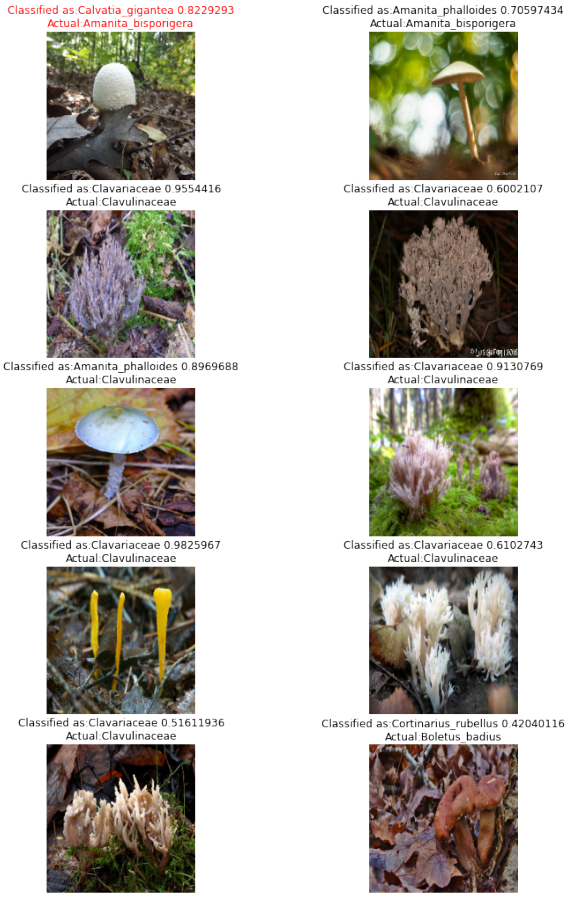

Figure 8: Misclassifications on our dataset. A poisonous species is classified as safe species with a probability of 0.82.

prove the performance of our model is take advantage of the fact that certain mushroom species are found in certain locations across the world and feed in the location of the mushroom species as a input to the neural network as well.

Our experiments also suggests that simpler models can be used without sacrificing performance. MobilenetV2 has the greatest APM metric of 0.217 and is tied in accuracy with Inception-RenNet-V2 at 0.76 on our dataset. MobilenetV2 also achieves a top 1 Accuracy of 74.7% on the Imagenet dataset with 1000 classes which is on par with other models such as VGG-19 (74.5%), Resnet50 (72.1%) (PaperswithCode). This suggests that our results will still hold for the task of mushroom classification if extended to a larger number of classes.

Our model matches the performance of another implementation of the mushroom classification problem using CNN (Preechasuk et al. 2019). This implementation achieves 74% accuracy on 45 different classes of mushrooms whereas our model achieves 76% accuracy. It is worthwhile to note that the model used in this study is just 6 layers deep with much less parameters than MobilenetV2 which suggests we can still downsize the model without much loss in performance.

We now discuss the limitations of our approach. The dataset used contained only 100 images from each mushroom class which is small for an image classification task. It would be interesting to see if the model's performance will increase if we increase the amount of data presented to it. The biggest problem that plagues neural networks is adversarial attacks. An adversarial attack is when a an input is imperceptibly altered to give the wrong output which would otherwise have been correctly classified. Neural networks are prone to such attacks and we need to make this model robust. This paper looks into some common defenses against adversarial attacks (Goel 2020) which can be used in our implementation.

Misclassifying a poisonous species as a safe one can be a potentially fatal mistake and more work needs to be done to make the model is robust and safe to use.

This model was trained and tested on the Google Colab environment with a Tesla T4 GPU and its performance on a platform with less computational power is yet to be tested. Mushroom foragers usually do not have access to large computational power and are often off the grid without access to the Internet. Therefore it would be useful to see how this model will perform on the mobile device.

Conclusion

We began the project by attempting to automatically classify different species of mushrooms. There are many species of mushrooms and some are poisnous. If we can find a way to automatically classify them correctly, it would be of great use to mushroom foragers worldwide. CNNs are the state of the art when it comes to image classification and it was natural choice for our task.

We then hypothesized that MobileNetV2 will be a good choice of CNN for our task due to its small size and good performance on the Imagenet dataset. MobilnetV2 uses bottlneck layers and a residual connection to help reduce its size without sacrificing performance.

We then chose 10 classes of mushroom species and scrapped the popular image hosting website Flickr to create our dataset of 1000 images with 100 form each class. 10 images from each class was set aside as a test set and the rest were used for training/classification. Images were rotated and flipped randomly for data augmentation.

The models were trained on our dataset and the base models were unfrozen and fine tuned on our dataset. MobilenetV2 and Inception-ResNet-V2 came out on top with an accuracy of 0.76 on our test dataset. MobileNetV2 model came out on top when to take model parameter size into account with a APM(accuracy per million parameter) of 0.217. MobileNetV2 was also the fastest model which classified 100 images on 1.256 seconds on the Google Colab environment with a Tesla T4 GPU.

Some possible directions this project can take are:-

1. Increase the number of classes and images under each class
2. Include location,texture,smell of the mushroom in our neural network

3. Experiment with downsizing the model and testing its performance on mobile devices

References

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions.

Dong, J., and Zheng, L. 2019. Quality classification of enoki mushroom caps based on cnn. In *2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC)*, 450–454.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* 12(null):2121–2159.

Goel, A. 2020. An empirical review of adversarial defenses.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015a. Deep residual learning for image recognition.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015b. Deep residual learning for image recognition.

Jocher, G. 2020. ultralytics/flickr_scraper: Release v1.

Kingma, D. P., and Ba, J. 2017. Adam: A method for stochastic optimization.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Ottom, M. A., and Alawad, N. A. 2019. Classification of mushroom fungi using machine learning techniques. *International Journal of Advanced Trends in Computer Science and Engineering* 8:2378–2385.

PaperswithCode. Papers with code - imagenet benchmark (image classification).

Preechasuk, J.; Chaowalit, O.; Pensiri, F.; and Visutsak, P. 2019. Image analysis of mushroom types classification by convolution neural networks. In *Proceedings of the 2019 2nd Artificial Intelligence and Cloud Computing Conference*, AICCC 2019, 82–88. ACM.

Sandler, M.; Howard, A. G.; Zhu, M.; Zhmoginov, A.; and Chen, L. 2018. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR* abs/1801.04381.

Sandra Sinno-Tellier, Chloé Bruneau, J. D. C. G. A. V. J. B. 2019. National surveillance of food poisoning by fungi: assessment of cases reported to the network of poison control centres from 2010 to 2017 in metropolitan france. *Bull Epidémiol Hebd* 33:666–678.

Simonyan, K., and Zisserman, A. 2015a. Very deep convolutional networks for large-scale image recognition.

Simonyan, K., and Zisserman, A. 2015b. Very deep convolutional networks for large-scale image recognition.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2014. Going deeper with convolutions.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015. Rethinking the inception architecture for computer vision.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning.

Tijmen Tieleman, G. H. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks.