Knot Recognition Systems

Anastasia Pupo
Khoury College
Northeastern University
Los Angeles, CA, USA
pupo.a@northeastern.edu

***Abstract*— Object recognition, especially of visually similar objects such as different types of knots, can pose a significant challenge. This paper investigates the effectiveness of convolutional neural networks (CNNs) architectures along with transfer learning with a pre-trained ResNet50 model for classifying 10 common climbing knots. The models are trained and evaluated on a publicly available image dataset from Kaggle to measure each models accuracy in distinguishing between various knot types. The goal is to train a model capable of reliably predicting knot types.**

I. Introduction

The task of this project is to use deep learning to recognize 10 different types of knots. These knots are commonly used for tasks like climbing, tying things together, and more. They can be helpful in wilderness settings, but also in everyday situations such as tying a load to a car or for fishing. These knots as well are used on the job by fishermen and firemen, making them a useful and practical skill to know.

Creating an application that can recognize knots may help people in learning this skill. Like any other skill, it also takes time to learn and remember how to tie some practical knots. Sometimes, a knot might be tied incorrectly, and it could be helpful to have an application that can detect verify if the knot was tied correctly. For this type of application, high accuracy would be very important, especially because in activities like climbing, tying a knot wrong can lead to serious injury. That said, it's important to note that deep learning predications should never be the only thing relied on for safety critical tasks.

Knots have also been studied as a mathematical problem in the field of knot theory, which is a complex topic [3]. However, this paper's focus will be on recognizing 10 specific knots with use of more common convolutional neural network architectures.

Convolutional neural networks (CNNs) are useful for identifying objects, and they are used in this project. Previous work by Joseph Cameron, who created a Kaggle dataset of knot images and wrote a dissertation on the topic, showed it is possible to classify these 10 knots utilizing deep learning [7]. In this paper, a CNN model will be trained to classify knots, and transfer learning with a pre-trained ResNet50 model will also be applied to compare between models.

II. Related Work

This section presents three peer-reviewed papers.

A. *A review of convolutional neural networks in computer vision*

This paper relates to this project because it discusses how convolutional neural networks (CNNs) are used in image classification, which is the method being used to recognize the different types of knots. It covers the basics of CNNs and includes updates from recent research up to early 2024. The paper shows that CNNs usually perform better than traditional computer vision techniques for classification tasks. It also points out some of the challenges with deep CNNs, such as being hard to interpret, needing a lot of memory, and being time-consuming to train. These issues are especially important when considering deployment on mobile devices, where resources are limited. The paper also discusses the difficulty of choosing the right hyperparameters and mentions how meta-heuristic algorithms can help. Finally, it highlights the importance of having a strong understanding of CNNs for building and optimizing models effectively [1].

B. *Evaluation of Transfer Learning Methods for Wood Knot Detection*

This paper conducted an experiment comparing four neural networks using transfer learning and training from scratch, along with eight different optimizers, resulting in a total of 64 classifiers to evaluate performance in wood knot classification. The authors mentioned how computer vision has made big improvements in areas like fire detection and tracking, so they believed there was reason to explore its potential for detecting wood knots as well. Optical grading systems are expensive and if a method of deep learning could help aid with the detection instead, then it would result in cost savings [2].

Even though the focus of this paper is on wood knots, the paper is still relevant to this project since it provides insight into working with a small dataset (80 wood knot images and 400 background images for training). This project also utilizes transfer learning with ResNet50. The paper noted that ResNet50 has fewer filters and is less complex than VGG. Their setup used 256x256 pixel images, 50 training epochs, and tested multiple optimizers. The optimizer Adam gave the highest average accuracy overall in their report, while Nadam performed best specifically with pretrained ResNet50 [2].

With this information, it helped or further supported what pre-trained neural network and optimizer to choose to use in this project. As well since in this paper it was determined that training the model from scratch versus utilizing a pre-trained model did not differ significantly between accuracy, at least for wood knots classification, this also led this project to focus on solving for a CNN that recognizes the different knots well.

C. *Geometric learning of knot topology*

This paper investigates how to use neural networks to attempt solving problems in knot theory based on geometry. Knot theory is the study of mathematical knots where the knots ends are joined to form a closed loop. Though this project contains images where the knots have free ends, there are knots like the square knot which can be converted to closed loop knots. This paper provides insight to learn more about the challenges of identifying knots and how knot topology such as number of crossing could aid in classification. The authors of this work stated the model in the paper was able to achieve 95% or above accuracy [3].

The paper relates to the project since it also looks at knots. This was also the only paper found besides Joseph Cameron's dissertation that mentioned any kind of common knot like the square or granny knot since many papers related to knots focus on closed loop mathematical knots and determining whether knots are different or duplicates since with many loops and crossing it becomes harder to recognize.

III. Dataset

The dataset utilized to train the models was obtained from Kaggle and is called 10Knots. It consists of the following ten different knots: the Alpine Butterfly Knot, the Bowline Knot, the Clove Hitch, the Figure-8 Knot, the Figure-8 Loop, the Fisherman's Knot, the Flemish Bend, the Overhand Knot, the Reef Knot, and the Slip Knot. There are 144 images per knot, making a total of 1,440 images.

For each knot, the 144 images consisted of variations across four different z-axis rotations, three lighting conditions, three tension levels, and two different backgrounds. View Fig. 1. for examples.

IV. Methods/Experimental Design

The 10Knots dataset form Kaggle will be utilized in training the model. Of the 1440 images, 80% of the images will be placed in the training dataset to help train the model. 10% will be utilized for validation during the training process. The remaining 10% will be utilized for the test dataset to test the model after training.

All images were resized to 256 x 256.

There were 4 total models trained:

1. Model 1: A model architecture of codebasics [8].
2. Model 2: A model architecture of the small convolution layer in Cameron's dissertation [7].
3. Model 3: A model architecture of the medium convolution layer in Cameron's dissertation [7].
4. Model 4: A model architecture of the transfer layer on pre-trained ResNet model [5].

Diagrams of each model's architecture shown in Table 1. are included at the end of this report

Each model is larger than the prior. For example, model 1 is the smallest as in has the lowest parameters.

Epochs were set to 50, and batch to 32.

Data augmentation performed where new images were generated to have more variation in zoom, rotation, translation/shift, and orientation of vertical and horizontal.

V. Experiments and Results

A. *Transfer Learning with ResNet50 Results*

When the model that was pretrained with ResNet50, 50 epochs were found to be too much since by epoch 5, the training and validation accuracies both surpassed 95% accuracy. It was decided to limit the training to 11 epochs where accuracies at this epoch was nearly 100% or very close. When testing the test dataset, the model had an accuracy of 99.49% and loss of 2.86%. In figure 1, there are 9 examples of images from the test set and the model's predictions. For these 9 images, the model was able to predict the class of each knot correctly.

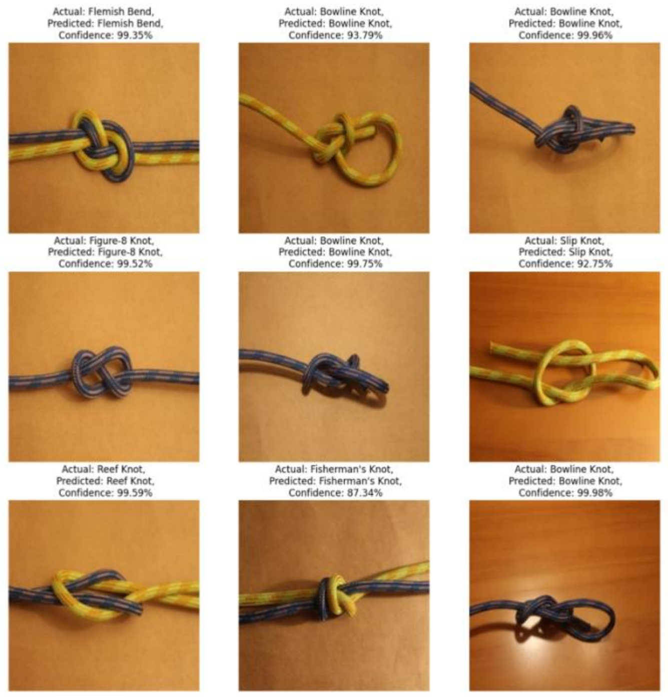


Fig. 1. Recognition of 9 images in the test dataset. Model used was pre-trained on ResNet50. All knots were classified correctly where actual class equaled predicted class. Confidence percentages were also provided.

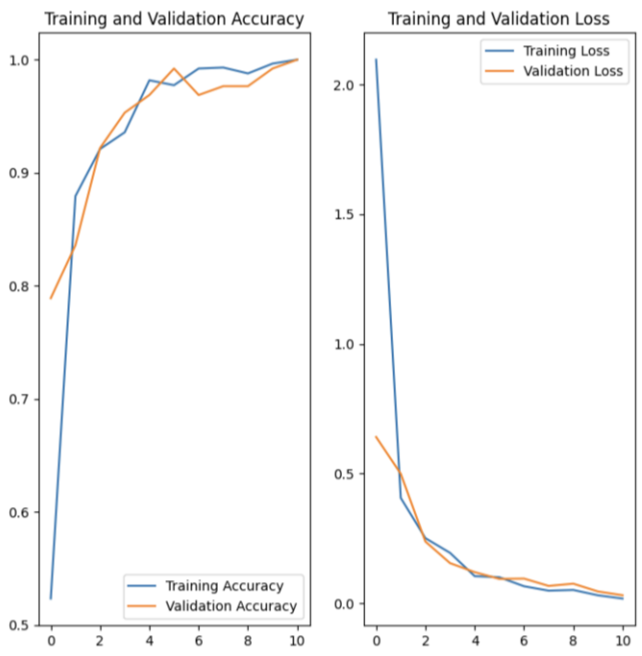


Fig. 2. Training and Validation Accuracy charts for model pre-trained on ResNet50. Training and validations accuracy have increased whiles training and validation loss decreased.

B. Training Convolutional Networks with 10Knots dataset Results

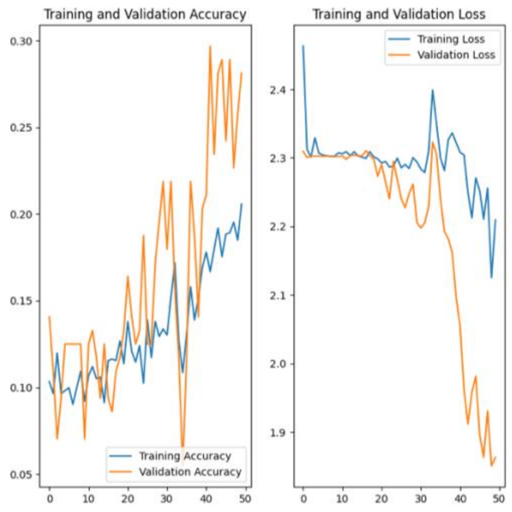


Fig.3. Model 3 training and validation accuracy and loss charts.

As can be noticed in Fig.3., there is low accuracy in both training and validation. The validation accuracy peaks at close to 30% and validation loss decreases to under 1.9, whilst training loss is over 2.1.

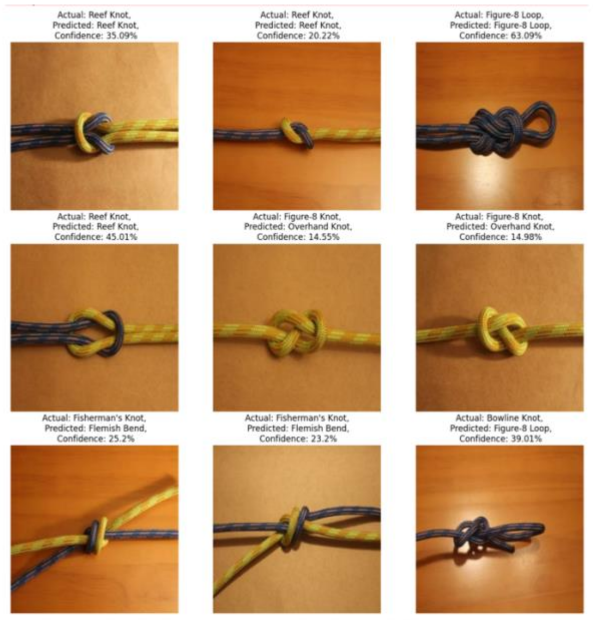

Actual: Reef Knot, Predicted: Reef Knot, Confidence: 35.09%
Actual: Reef Knot, Predicted: Reef Knot, Confidence: 20.22%
Actual: Figure-8 Loop, Predicted: Figure-8 Loop, Confidence: 63.09%
Actual: Reef Knot, Predicted: Reef Knot, Confidence: 45.01%
Actual: Figure-8 Knot, Predicted: Overhand Knot, Confidence: 14.55%
Actual: Figure-8 Knot, Predicted: Overhand Knot, Confidence: 14.98%
Actual: Fisherman's Knot, Predicted: Flemish Bend, Confidence: 25.2%
Actual: Fisherman's Knot, Predicted: Flemish Bend, Confidence: 23.2%
Actual: Bowline Knot, Predicted: Figure-8 Loop, Confidence: 39.01%

Fig.4. Result of Model 3 predictions. Not all knots are classified correctly.

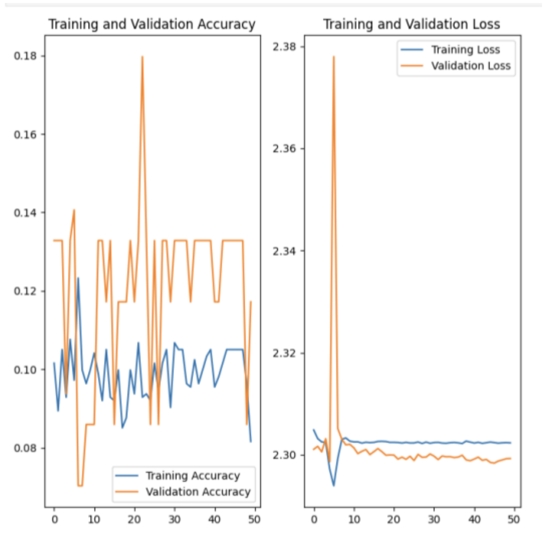


Fig.5. Model 1 training and validation accuracy and loss charts.

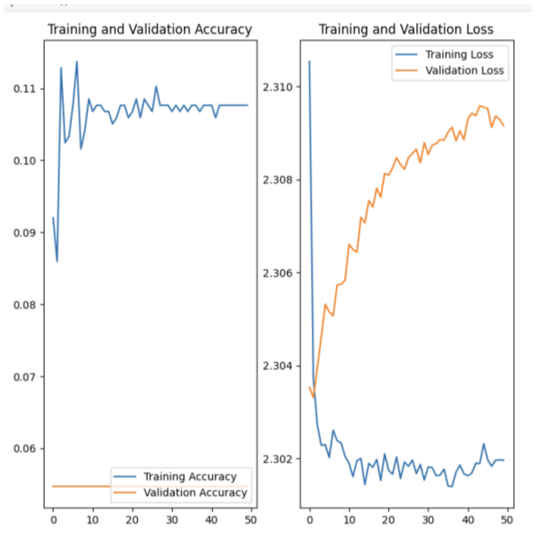


Fig.6. Model 2 training and validation accuracy and loss charts. There is an error with validation.

Model 3 performed better than Model 1 and Model 2, while Model 2 performed the worst. Both Model 2 and Model 1 appear to have issues with validation. This may be due to a change in learning rate, since previously the learning rate was 0.0003, and was changed to the default for the Adam optimizer. There is possibility as well that there is overfitting since there is a large difference between the values of the training and validation data, especially in Fig.6.

VI. Discussion and Summary

Transfer learning with the pretrained ResNet50 model performed the best in recognition. After training the convolutional neural networks and viewing low accuracy rates for each model, it could have been assumed that perhaps the transfer learning with ResNet50 may also not perform well. However, it had close to 100% accuracy and therefore would be a good model to predict these 10 knots. To improve the other models, having a larger dataset may help as well as including additional optimizations not implemented in this project. In the dissertation that first trained a CNN on the 10Knots dataset, the highest accuracy was 77.5% when utilizing the image classifier in an iOS application[7]. The peer-review article "A review of convolutional networks in computer vision" has new methods highlighted that could help with improving performances in classification. As well the "Geometric learning of knot topology" as well may help improve the models by having more weight towards geometry and topology.

Overall, a model was able to found that identified the different types of 10 knots in 10Knots dataset very well. There can still be further improvements and experimentation done. For example, a next step can be to capture images of knots and view how it performs since the 10Knots dataset is quite uniform compared to real life pictures where there could be obstructions or other aspects of images that could cause some complexity in the process object identification and classification.

References

[1] Zhao, X., Wang, L., Zhang, Y. *et al.* A review of convolutional neural networks in computer vision. *Artif Intell Rev* 57, 99 (2024). https://doi.org/10.1007/s10462-024-10721-6

[2] Norlander, R., Grahn, J., Maki, A. (2015). Wooden Knot Detection Using ConvNet Transfer Learning. In: Paulsen, R., Pedersen, K. (eds) Image Analysis. SCIA 2015. Lecture Notes in Computer Science(), vol 9127. Springer, Cham. https://doi.org/10.1007/978-3-319-19665-7_22

[3] Sleiman, J. L., Conforto, F., Fosado, Y. a. G., & Michieletto, D. (2023). Geometric learning of knot topology. *Soft Matter*, *20*(1), 71–78. https://doi.org/10.1039/d3sm01199b

[4] *10Knots*. (2018, March 24). Kaggle. https://www.kaggle.com/datasets/josephcameron/10knots

[5] https://github.com/nachi-hebbar/Transfer-Learning-ResNet-Keras/blob/main/ResNet_50.ipynb

[6] Joseph Manfredi Cameron. 10 Knots Dataset: The Comprehensive Dataset of Knots. https://www.kaggle.com/datasets/josephcameron/10knots

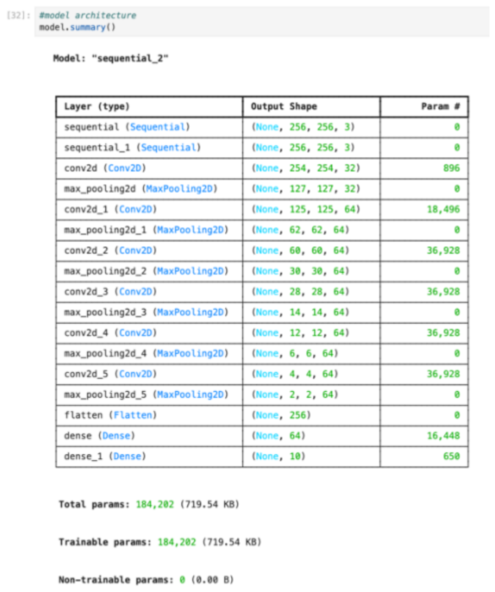
[7] https://github.com/JoeCameron1/IndividualProject/blob/master/Dissertation/dissertation.pdf

[8] https://www.youtube.com/playlist?list=PLeo1K3hjS3utJFNGyBpIvjWgSDY0eOE8S

Table 1. Model Architectures.

Model Architectures

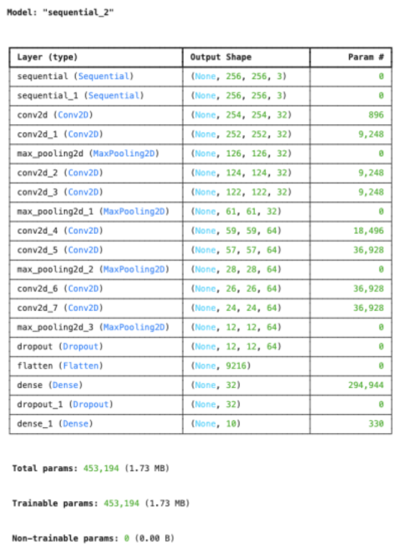
Model 1 Architecture:



```
[32]: #model architecture
      model.summary()
```

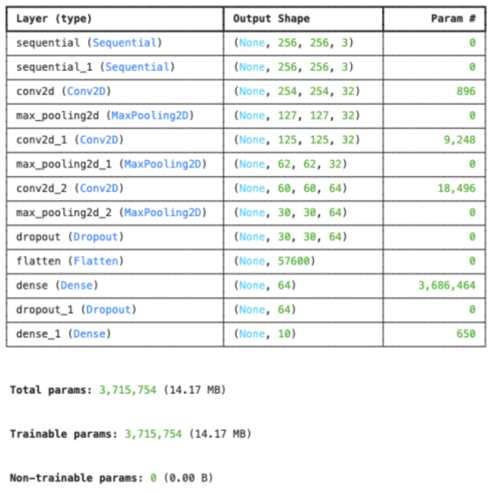
```
Model: "sequential_2"

Layer (type)                     Output Shape              Param #
sequential (Sequential)          (None, 256, 256, 3)             0
sequential_1 (Sequential)        (None, 256, 256, 3)             0
conv2d (Conv2D)                  (None, 254, 254, 32)          896
max_pooling2d (MaxPooling2D)     (None, 127, 127, 32)            0
conv2d_1 (Conv2D)                (None, 125, 125, 64)       18,496
max_pooling2d_1 (MaxPooling2D)   (None, 62, 62, 64)              0
conv2d_2 (Conv2D)                (None, 60, 60, 64)         36,928
max_pooling2d_2 (MaxPooling2D)   (None, 30, 30, 64)              0
conv2d_3 (Conv2D)                (None, 28, 28, 64)         36,928
max_pooling2d_3 (MaxPooling2D)   (None, 14, 14, 64)              0
conv2d_4 (Conv2D)                (None, 12, 12, 64)         36,928
max_pooling2d_4 (MaxPooling2D)   (None, 6, 6, 64)                0
conv2d_5 (Conv2D)                (None, 4, 4, 64)           36,928
max_pooling2d_5 (MaxPooling2D)   (None, 2, 2, 64)                0
flatten (Flatten)                (None, 256)                     0
dense (Dense)                    (None, 64)                 16,448
dense_1 (Dense)                  (None, 10)                    650

Total params: 184,202 (719.54 KB)

Trainable params: 184,202 (719.54 KB)

Non-trainable params: 0 (0.00 B)
```

Model 2 Architecture:



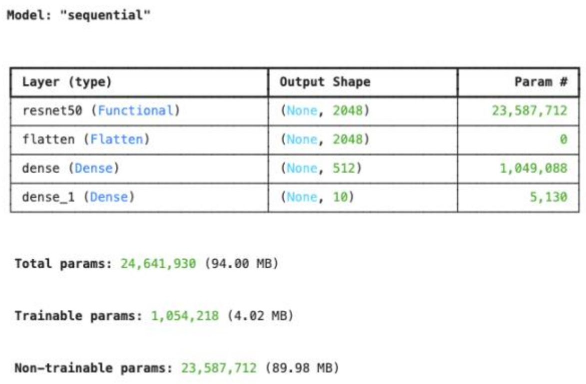
```
Model: "sequential_2"

Layer (type)                     Output Shape              Param #
sequential (Sequential)          (None, 256, 256, 3)             0
sequential_1 (Sequential)        (None, 256, 256, 3)             0
conv2d (Conv2D)                  (None, 254, 254, 32)          896
conv2d_1 (Conv2D)                (None, 252, 252, 32)        9,248
max_pooling2d (MaxPooling2D)     (None, 126, 126, 32)            0
conv2d_2 (Conv2D)                (None, 124, 124, 32)        9,248
conv2d_3 (Conv2D)                (None, 122, 122, 32)        9,248
max_pooling2d_1 (MaxPooling2D)   (None, 61, 61, 32)              0
conv2d_4 (Conv2D)                (None, 59, 59, 64)         18,496
conv2d_5 (Conv2D)                (None, 57, 57, 64)         36,928
max_pooling2d_2 (MaxPooling2D)   (None, 28, 28, 64)              0
conv2d_6 (Conv2D)                (None, 26, 26, 64)         36,928
conv2d_7 (Conv2D)                (None, 24, 24, 64)         36,928
max_pooling2d_3 (MaxPooling2D)   (None, 12, 12, 64)              0
dropout (Dropout)                (None, 12, 12, 64)              0
flatten (Flatten)                (None, 9216)                    0
dense (Dense)                    (None, 32)                294,944
dropout_1 (Dropout)              (None, 32)                      0
dense_1 (Dense)                  (None, 10)                    330

Total params: 453,194 (1.73 MB)

Trainable params: 453,194 (1.73 MB)

Non-trainable params: 0 (0.00 B)
```

Model 3 Architecture:



```
Layer (type)                     Output Shape              Param #
sequential (Sequential)          (None, 256, 256, 3)             0
sequential_1 (Sequential)        (None, 256, 256, 3)             0
conv2d (Conv2D)                  (None, 254, 254, 32)          896
max_pooling2d (MaxPooling2D)     (None, 127, 127, 32)            0
conv2d_1 (Conv2D)                (None, 125, 125, 32)        9,248
max_pooling2d_1 (MaxPooling2D)   (None, 62, 62, 32)              0
conv2d_2 (Conv2D)                (None, 60, 60, 64)         18,496
max_pooling2d_2 (MaxPooling2D)   (None, 30, 30, 64)              0
dropout (Dropout)                (None, 30, 30, 64)              0
flatten (Flatten)                (None, 57600)                   0
dense (Dense)                    (None, 64)              3,686,464
dropout_1 (Dropout)              (None, 64)                      0
dense_1 (Dense)                  (None, 10)                    650

Total params: 3,715,754 (14.17 MB)

Trainable params: 3,715,754 (14.17 MB)

Non-trainable params: 0 (0.00 B)
```

Model 4 Architecture:



```
Model: "sequential"

Layer (type)                     Output Shape              Param #
resnet50 (Functional)            (None, 2048)           23,587,712
flatten (Flatten)                (None, 2048)                    0
dense (Dense)                    (None, 512)             1,049,088
dense_1 (Dense)                  (None, 10)                  5,130

Total params: 24,641,930 (94.00 MB)

Trainable params: 1,054,218 (4.02 MB)

Non-trainable params: 23,587,712 (89.98 MB)
```