HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 1 (2023)

Application of Convolutional Neural Networks to Classify Ambiguous Categories

Arjun Rai

Carroll Senior High School

ABSTRACT

Neural networks are often used for classifying images of specific objects, people, animals, and other objects of interest because of their ability to find particular patterns for categories. In this paper, we apply a Convolutional Neural Network (CNN) to classify images from a dataset of 4 semi-ambiguous classes and compare the scores of different architectures of neural networks and how different preprocessing techniques can affect their performance.

1 Introduction

Abstract ideas can be interpreted in an uncountable number of ways due to their lack of explicit characteristics or physical features. As a result, any individual can interpret or visualize an abstract idea differently. This stems from every individual being unique, having a different background, culture, and upbringing which all influence their understanding of abstract ideas. However, this perception does not extend to neural networks which aim to try and replicate human cognition or pattern recognition in a generalizable manner. In this paper we attempt to use deep convolutional neural networks to categorize images that represent these abstract ideas to see if networks can identify features in high dimensional data that allow them to distinguish between multiple ambiguous categories.

1.1 Data

The data used in this paper, *Google Scraped Image Dataset*, was taken from Kaggle, an online data provider. [1] This data set is made up of images and has four categories: art/culture, architecture, travel and adventure, and food and drink. There are around 32,000 images, with approximately eight thousand in each class, making it a relatively balanced data set. The photos are in various formats, sizes, and sources. Supervised learning was the training methodology used as the images were already labeled. Section 2 will cover the methodology used to preprocess the images and train the models. Section 3 will show the results of different architectures of neural networks on the data, and section 4 will conclude the findings.

2 Methodology

2.1 Preprocessing

The image categories are abstract, but they all contain distinct features. Since the images were of various widths and heights, as seen in Figure 1, the images were standardized into a single image size.

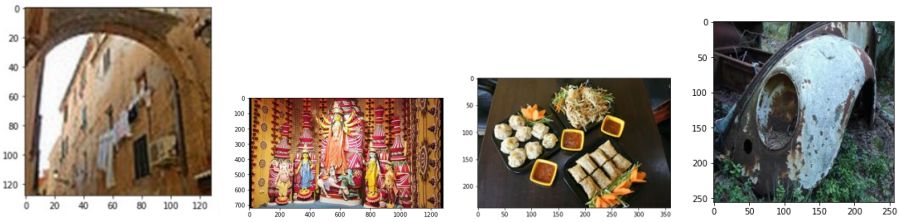

Figure 1. From left to right: Architecture, Art/Culture, Food and Drink, and Travel and Adventure.

The minimum width and height of the images were found to find an image size that works for all images. However, one outlier image was much smaller than the rest, so it was removed. The images were then cut down to 16 thousand images (four thousand in each category) to save memory and resized to two different sizes: 64x64 and 100x100. The preprocessed images were also saved to avoid re-processing.

Finally, the images were mapped to their labels as follows:

0: Architecture
1: art/culture
2: Food and Drinks
3: Travel and Adventure

2.2 Proposed Tests

To test how the different preprocessing and neural network architectures would affect the loss and accuracy, the following was done:

- Changed the size of the image for the specific test.
- Changed the network architecture by adding more layers, changing the number of dense nodes, adding kernel regularization (lambda), changing the size of the 2D convolutional filters, and changing various other parameters.

All models were trained until the categorical cross-entropy loss and the loss converged. The final validation loss, loss, validation accuracy, and training accuracy were taken.

3 Results

3.1 Single Convolutional Layer Architecture

The first architecture applied to the dataset used all 8000 images in each class with the images resized and normalized to 100x100. A convolutional layer is used in this problem and placed first because of its ability find the unique components of the images to help the rest of the neural network. [2] Instead of connecting every value with its own node, convolutional layers connect regions to nodes, making them more efficient. [3] The model had a architecture as seen in figure 2.

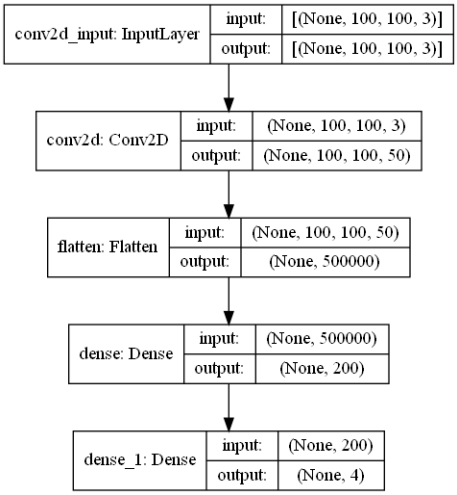

Figure 2. Neural Network architecture of first model used.

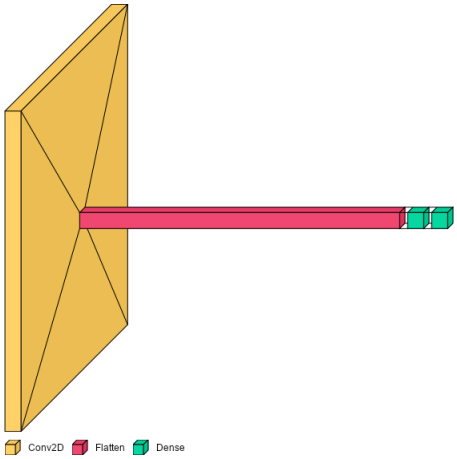

Figure 3. Visualization of first Neural Network architecture.

Table 1. The statistics of the first model on the test set.

Accuracy	Validation Accuracy	Loss	Validation Loss
0.99	0.58	0.0164	3.765

3.2 Single Convolutional Layer Architecture Continued

In the following architecture, all the images were used in the 100x100 size. Because the neural network was overfitting, the number of dense nodes was reduced to try and minimize the overfitting. Reducing the number of nodes or layers makes the network less complex.

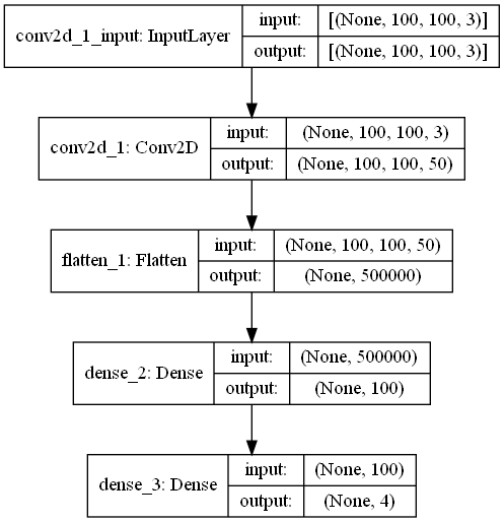

Figure 4. Neural Network architecture of second model used.

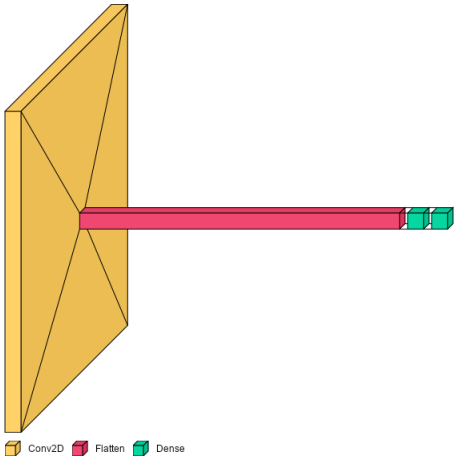

Figure 5. Visualization of second Neural Network architecture.

Table 2. The statistics of the second model on the test set.

Accuracy	Validation Accuracy	Loss	Validation Loss
0.99	0.61	0.019	2.9

3.3 Single Convolutional Layer Architecture with Max Pooling

Since changing the number of dense nodes had little effect, a max pooling layer was added. Max pooling reduces the complexity of the data for the layers after it by taking the maximum value in a sub-region and creating a smaller matrix based off of the maxes. [3] Due to the limited amount of memory on the computer used for training, the size of the images was decreased to 64x64 for this model.

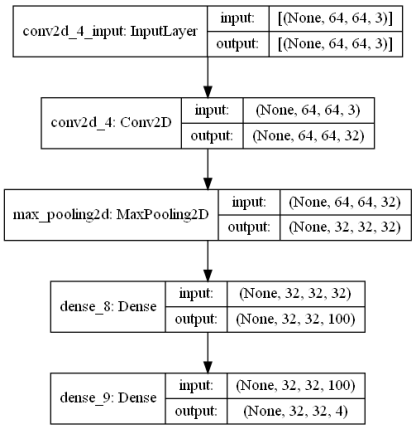

Figure 6. Neural Network architecture of third model used.

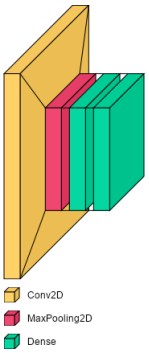

Figure 7. Visualization of third Neural Network architecture.

Table 3. The statistics of the third model on the test set.

Accuracy	Validation Accuracy	Loss	Validation Loss
0.73	0.67	0.6714	0.8901

3.4 Two Convolutional Layer Architecture with Max pooling and Batch Normalization

Next, the model was made more complex with another set of max pooling layers, convolutional layers, and batch normalization layers. The number of images were also reduced to 4000 in each class due to computational limits. Dropout layers were also added to the dense layers to minimize overfitting. Batch normalization layers were added because they balance the data distribution, which speeds up training and makes the model more predictive as it does not over fit to a single class.[4] The layers are placed in the order shown as the max pooling layer reduces the complexity of the data that was processed by the convolutional layers, and the batch normalization layer normalizes the data for the next convolutional layer. After the convolutional layers, the flatten layer was used to convert the 2D matrix into a 1D matrix so that the dense nodes could be used.

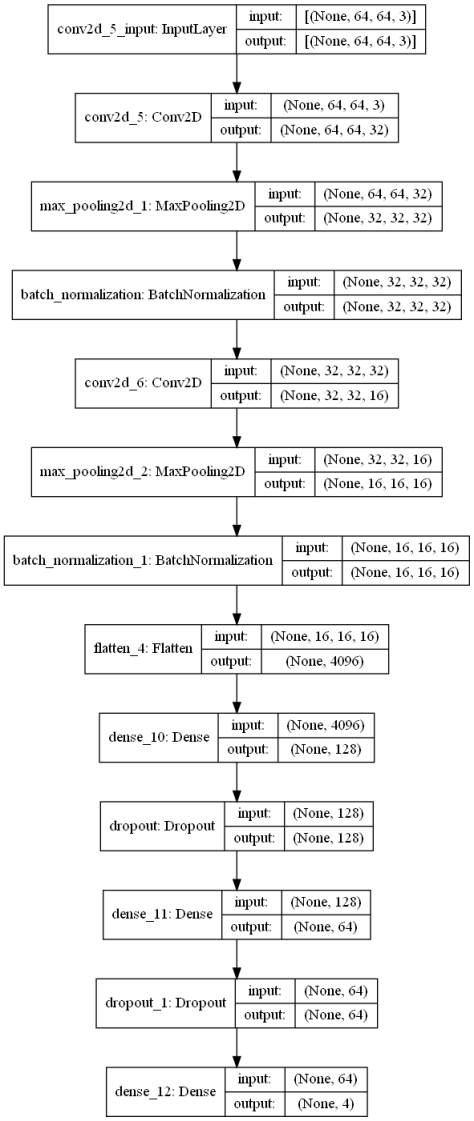

Figure 8. Neural Network architecture of fourth model used.

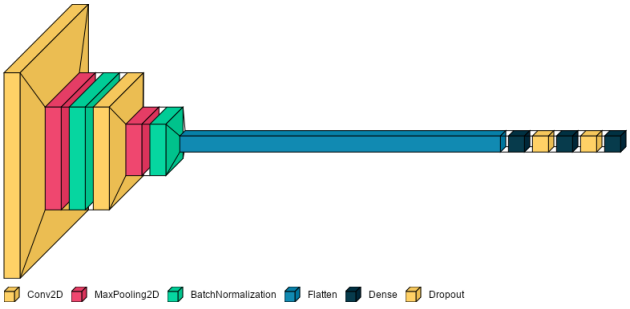

Figure 9. Neural Network architecture of fourth model used.

Table 4. The statistics of the fourth model on the test set.

Accuracy	Validation Accuracy	Loss	Validation Loss
0.94	0.73	0.1558	1.6432

3.5 Two Convolutional Layer Architecture with Max pooling, Batch Normalization, and Kernel Regularization

Since there was still overfitting, L1 kernel regularizers were added to keep the weights small. Larger weights mean that the network is overcompensating to get the training samples correct. The size of the images was also increased back up to 100x100 to give the model more information.

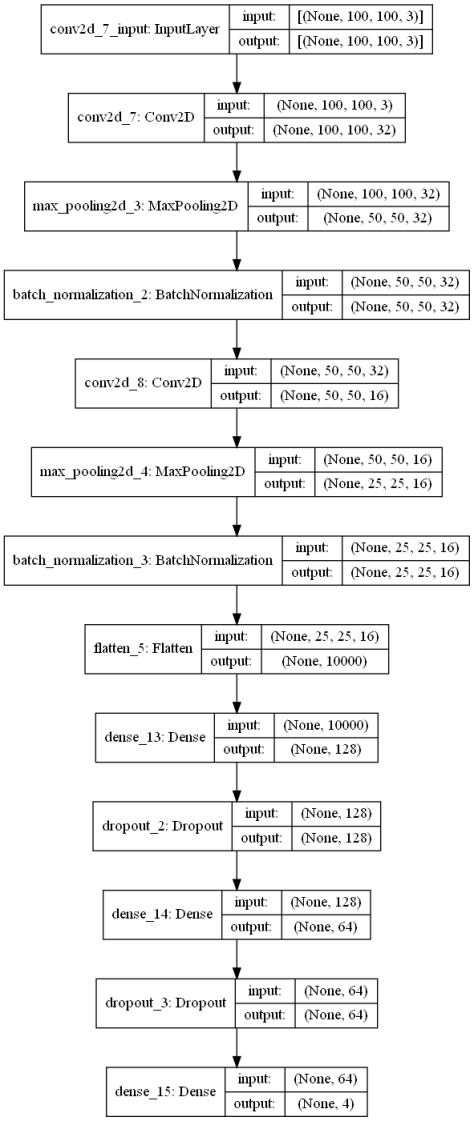

Figure 10. Neural Network architecture of fifth model used.

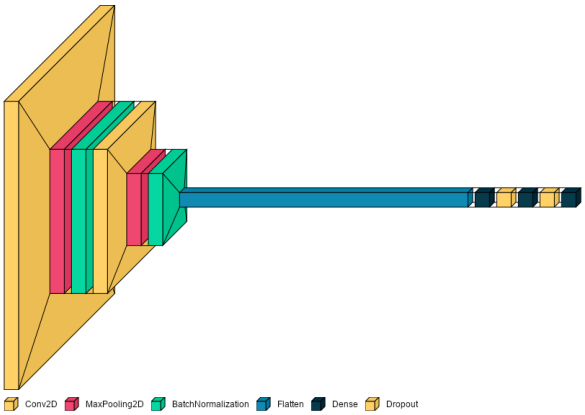

Figure 11. Visualization of fifth Neural Network architecture.

Table 5. The statistics of the fifth model on the test set.

Accuracy	Validation Accuracy	Loss	Validation Loss
0.9121	0.9319	3.0661	2.8861

Since this model had the most parameters to tune, kernel regularizers, and dropout layers, the model could be trained for more epochs and reach a validation accuracy equal to or greater than the training accuracy. The classification of images of ambiguous classes worked with the dataset, but with such general categories and little data compared to all possible examples, it does poorly with real-world data. Here are some pictures I took and some images from the internet that the model was tested against:

Figure 12. Example of labels predicted on new images.

The network most likely misclassified these images because of the lack of data compared to the number of different images that could be associated with the abstract class.

4 Conclusion

In this paper we tested multiple CNN architectures for classification of images from ambiguous categories. The architecture that worked the best consisted of two sets of 2D convolution layers, max pooling, and batch normalization layers with 3 layers of dense nodes. It classified the images with their abstract concept labels with an accuracy of 93.19%. Although images in the categories of art/culture and architecture could be classified differently by different individuals, the neural network classified more items as art/culture than architecture. Further, the images in the category food & drink would likely be classified similarly by different people but the network fails to classify these images correctly. With such broad ideas in these classes, it would be interesting to see if deep neural networks could identify these abstract concepts more accurately with larger amounts data in the future.

References

[1] R. Google Scraped Image Dataset. (2022). Retrieved 27 August 2022, from https://www.kaggle.com/datasets/duttadebadri/image-classification

[2] Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. Neural computation, 29(9), 2352-2449. https://doi.org/10.1162/neco_a_00990

[3] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). Ieee. https://doi.org/10.1109/ICEngTechnol.2017.8308186

[4] Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization?. Advances in neural information processing systems, 31. https://doi.org/10.48550/arXiv.1805.11604