*Convolutional Neural Network for Multiclass Image Classification of Filipino Dishes)*

Simon Daniel Dela Cruz  
*Computer Science Department  
BSCS STEM – 3AB  
Technological University of the Philippines  
Ayala Blvd, Ermita, Manila, 1000 Metro Manila, Philippines*simondaniel.delacruz@tup.edu.ph

Joshuel Ernest Simbulan  
*Computer Science Department  
BSCS STEM – 3AB  
Technological University of the Philippines  
Ayala Blvd, Ermita, Manila, 1000 Metro Manila, Philippines*joshuelernest.simbulan@tup.edu.ph

Andrew James Tejerero  
*Computer Science Department  
BSCS STEM – 3AB  
Technological University of the Philippines  
Ayala Blvd, Ermita, Manila, 1000 Metro Manila, Philippines*andrewjames.tejerero@tup.edu.ph

Mark Zaired Ugay  
*Computer Science Department  
BSCS STEM – 3AB  
Technological University of the Philippines  
Ayala Blvd, Ermita, Manila, 1000 Metro Manila, Philippines*markzaired.ugay@tup.edu.ph

*Abstract*— This research aims to develop and evaluate a convolutional neural network (CNN) for the task of multiclass image classification of Filipino dishes. The increasing popularity of Filipino cuisine worldwide has made it necessary to have an automated system for identifying dishes. Traditional methods for dish classification such as manual annotation are time-consuming and prone to errors. With the advent of deep learning, it has become possible to train models that can accurately classify images of dishes.

The dataset used for training and testing the CNN consists of images of various Filipino dishes, with each image labeled according to its dish category. The dataset was collected by the researchers, and it contains a diverse set of images that represent the different dishes from different regions of the Philippines. The dataset was split into training and validation sets with a ratio of 80:20. The CNN architecture and hyperparameters were optimized through a series of experiments to achieve the highest possible classification accuracy. The researchers used TensorFlow and Keras, two popular open-source deep learning libraries, to develop and train the program. The program is written in Python, a popular programming language used in various data analytics related tasks. The performance of the developed CNN was then evaluated using standard metrics. The results of the evaluation demonstrate the effectiveness of the proposed CNN in accurately classifying images of Filipino dishes. The research contributes to the development of a model that can automate the task of identifying Filipino dishes from images, which can have practical applications in the food industry such as in food delivery apps and recipe search engines. The model can also be used in restaurant menus, food festivals, and other related applications. Furthermore, this research can also be used as a benchmark for other food classification tasks.

Keywords—convolutional neural network (CNN), multiclass image classification, Filipino dishes, deep learning, TensorFlow, Keras, Python Jupyter

1. INTRODUCTION

The increasing popularity of Filipino cuisine worldwide has made it necessary to have an automated system for identifying dishes. Traditional methods for dish classification such as manual annotation are time-consuming and prone to errors. Deep learning has made it possible to train models that can properly classify photographs of food. In this study, we propose to develop and assess a convolutional neural network (CNN) for the task of multiclass image classification of Filipino dishes. We will use a dataset of images of various Filipino dishes, with each image labeled according to its dish category. To attain the maximum classification accuracy feasible, the CNN architecture and hyperparameters will be refined through a series of trials. Standard performance measures will be used to assess how well the constructed CNN performs. The algorithm will be created and trained by the researchers using TensorFlow and Keras, two well-known open-source deep learning tools. Python, a popular programming language used in various data analytics related tasks. The study will aid in the creation of a model that can automatically recognize Filipino cuisine from photos. This model may find use in the food business, including in recipe search engines and food delivery apps.

1. RELATED LITERATURE

The latest generation of Convolutional Neural Networks (CNN) have achieved impressive results in challenging benchmarks on image recognition and object detection, significantly raising the interest of the community in these methods. Convolutional Neural Networks (CNNs) are analogous to traditional Artificial Neural Networks in that they are comprised of neurons that self-optimize through learning. Each neuron will still receive an input and perform an operation (such as a scalar product followed by a nonlinear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply [1].

Overfitting is a fundamental issue in supervised machine learning which prevents us from perfectly generalizing the models to well fit observed data on training data, as well as unseen data on testing set. Because of the presence of noise, the limited size of training set, and the complexity of classifiers, overfitting happens.[2]

Convolutional layer parameters focus around the use of learnable kernels. These kernels are usually small in spatial dimensionality, but spread along the entirety of the depth of the input. When the data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a 2D activation map.Convolutional layers are also able to significantly reduce the complexity of the model through the optimisation of its output. These are optimized through three hyperparameters, the depth, the stride and setting zero-padding. [3]

Main kernel initialization method used in deep learning, specifically in the initialization of the weights in a neural network is the he\_normal method. The He\_normal initialization method draws random samples from a normal distribution with a mean of 0 and a standard deviation of sqrt(2/n), where n is the number of input units in the weight tensor[4]. This initialization method is often used in convolutional neural networks, and it has been found to work well in practice.

TensorFlow is the most popular one among the plethora of deep learning libraries. In the field of deep learning, neural networks have achieved tremendous success and gained wide popularity in various areas, having multilayer perceptron, the convolutional neural network, and stochastic gradient descent as the most commonly used optimization method for neural network models.[5]

Convolutional Neural Networks (CNNs) have achieved impressive results in challenging benchmarks on image recognition and object detection. Convolutional layer parameters focus around the use of learnable kernels. These kernels are usually small in spatial dimensionality, but spread along the entirety of the depth of the input. When data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a 2D activation map. He\_normal method draws random samples from a normal distribution with a mean of 0 and a standard deviation of sqrt(2/n).

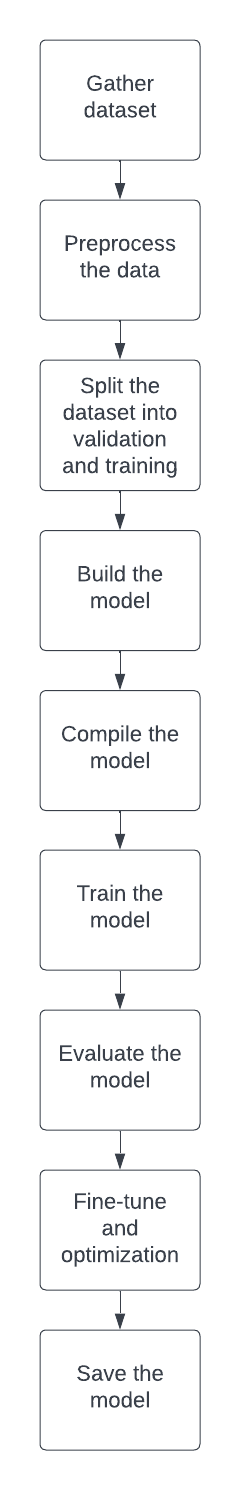
1. METHODOLOGY

The primary goal of this study is to create and train a CNN model with training and validation accuracy above the given benchmark of 75%. Achieving low training and validation loss is also a priority in the creation of the model.

|  |  |
| --- | --- |
| Table 1. IMAGE DATASET | |
| NAME OF DISH | SIZE OF CLASS |
| Adobo | 945 |
| Pinakbet | 951 |
| Sinigang | 979 |
| TOTAL SIZE | 2875 |

It is important to note that the size of the class (or category) must be roughly the same—or balanced, as imbalanced class size might result to the model being good at recognizing one class, but not the others. The dataset was collected manually through Google images. Preprocessing the data, ie resizing the data, was done through code.

*Image Classifier Model Development*



*Figure 1. Development Framework*

The methodology employed in this study involved utilizing a flowchart as a blueprint for the creation of the model. The initial step involved obtaining the necessary data through web scraping techniques. Subsequently, the data was preprocessed to eliminate any defective files. This was accomplished through a combination of manual review to identify and eliminate files that were inconsistent with the dataset, as well as implementing code to automatically detect and remove any unusable files. The dataset was split into validation and training with the ratio of 20:80. Training data is used to train a machine learning model to learn patterns in the data, while validation data is used to evaluate the model's performance on unseen data, giving an estimate of how well it will perform on new data. The model architecture followed the convention of three convolutional layers, and was later optimized through regularization techniques and a series of experimentation.

*Importing the Dataset*

In order to obtain the dataset, we employed Tensorflow's Keras method for loading image datasets from a directory. A validation split of 20% was utilized, with the validation set comprising 20% of the overall dataset. The image size was subsequently adjusted using the image\_size parameter to the appropriate dimensions.

*Text

Description automatically generated*



*Figure 2. Training Dataset*

Text

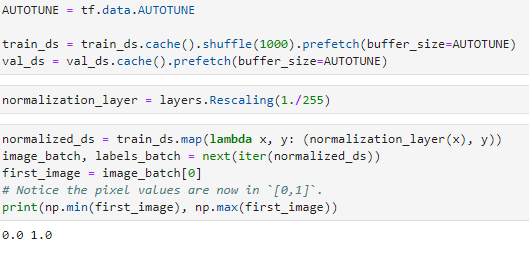
Description automatically generated



*Figure 2.1. Validation Dataset*

Figures 2.1 and 2.2 show the split of the data into two categories: training and validation. The seed parameter is used to randomly select which images go where.

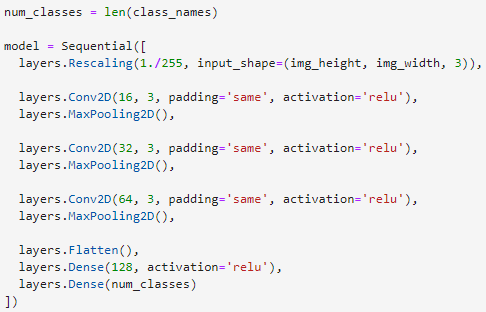
*Configure and Standardize the Dataset*

**

*Figure 2.2. Configuring and Standardizing the Dataset*

This code is caching, shuffling and prefetching the train\_ds and val\_ds datasets for the best performance during training.

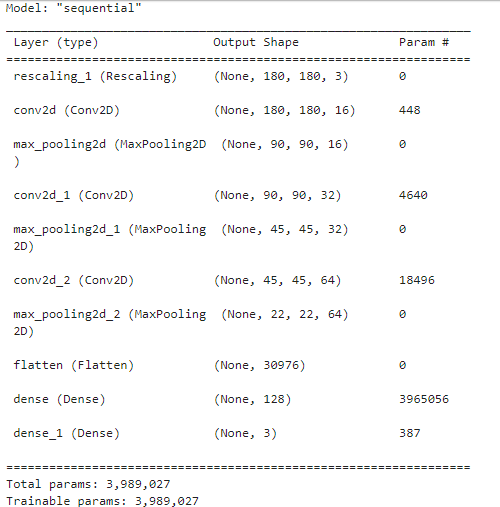
*Building the Model Architecture*



*Figure 3. Creating the Sequential model*

In the given sequential model, the researchers followed the convention of starting with two to three convolutional layers, maxpooling each layer to reduce the dimensionality of the image. This effectively reduces the spatial resolution of the image while retaining the most important information. Then, we used the flatten operation to convert the multi-dimensional output of the convolutional layers into a one-dimensional array, so that it can be input into the fully connected layers. Finally, we used dense layers, also known as fully connected layers to connect all of the neurons in the previous layer to the neurons in the current layer.

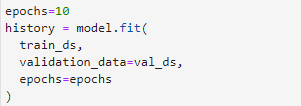




*Figure 3.1 model.summary()*

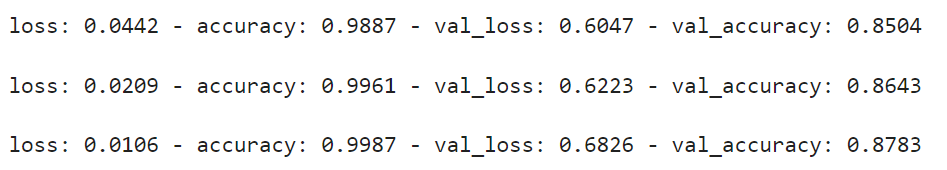
After we created the convolutional network architecture, we trained the model with the Filipino dish dataset.

Model.fit() *(Training the Model)*

**

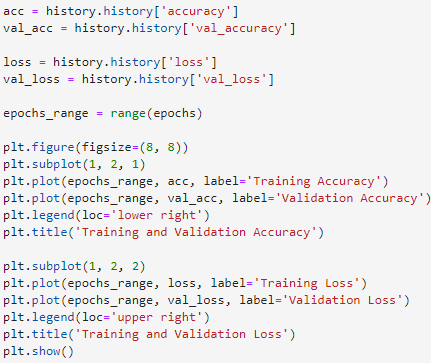
*Figure 3.2 model.fit()*

The CNN model architecture was trained utilizing the model.fit() method with the training and validation datasets, with a total of ten epochs specified. It is crucial to strike a balance in the number of epochs, as an insufficient number may result in inadequate training and an inability to accurately identify input images, while an excessive number may lead to overfitting.

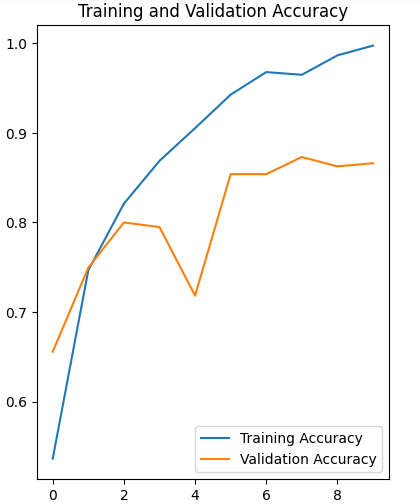


*Figure 4.1. Last three epochs*

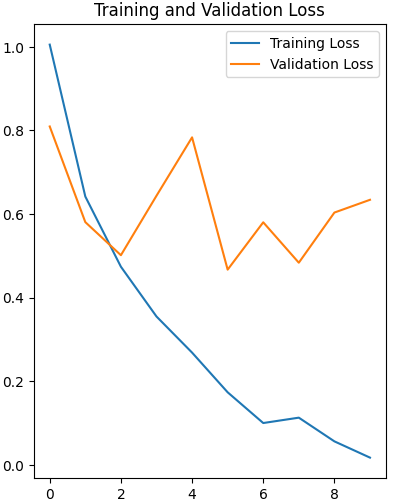
As shown in Figure 4, the loss continued to decrease up to 1%, accuracy was able to reach up to 99% while validation loss stagnated at around 60% and validation accuracy at 85%. To further visualize the result, we have used the following code:



*Figure 4.2. Result Visualization (code)*

**

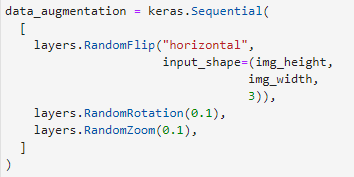
*Figure 4.3.a. Accuracy*



*Figure 4.3.b Loss*

While both the training and validation accuracy increased as time went by, we can observe that the loss and validation loss diverged significantly. This divergence is a common sign of overfitting. To prevent overfitting, we introduced techniques such as dropout and data augmentation.

*Prevent Overfitting*



*Figure 5. Data Augmentation Layers*

The RandomFlip layer randomly flips the input image horizontally. The input shape of the image is specified as (img\_height, img\_width, 3), where img\_height and img\_width are the dimensions of the image and 3 is the number of color channels (RGB). The RandomRotation layer randomly rotates the input image by a small amount (0.1 radians). The RandomZoom layer randomly zooms the input image by a small factor (0.1).

# Graphical user interface, text Description automatically generated with medium confidence

*Figure 5.1 New Sequential Model*

The new model architecture incorporates the data augmentation model created earlier, followed by a series of 3 convolutional layers, each paired with a batch normalization layer. These layers work together to extract features from the input image, and the batch normalization helps to stabilize and speed up the training process. Additionally, max pooling layers are used to reduce the spatial dimensions of the input and improve the model's ability to generalize to new data. A dropout layer is also included to prevent overfitting by randomly "turning off" some neurons during training. The output from these layers is then flattened and passed through two dense layers, the first one with 128 neurons and the second one with num\_classes neurons, with the second one is responsible for the classification of the input image. Overall, the architecture is designed to enhance the model's ability to classify images accurately.

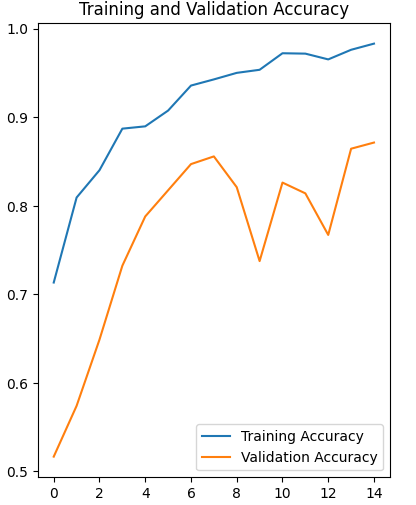
The new model was trained on 15 epochs, 5 more epochs than the previous one.

Text

Description automatically generated

*Figure 6. Last Three Epochs*

As shown in Figure 6, the loss continued to decrease up to 7%, accuracy was able to reach up to 99% while validation loss was able to reach 54% and validation accuracy at 87%. The following figure will further visualize the result.



*Figure 6.a. Accuracy*

*Chart, line chart

Description automatically generated*

*Figure 6.b. Loss*

The trend in Figure 6.a illustrates an increase in accuracy for both the validation and training sets. Figure 6.b demonstrates that the lines for validation and training loss remained consistent and did not deviate from each other.

1. RESULTS

As previously discussed in the preceding chapter, the optimized model was able to achieve improved results through the implementation of regularization techniques and other experimental methods.

In this section, the researchers will provide a comprehensive analysis of the model's performance by presenting standard evaluation metrics such as the confusion matrix and F1-score.

The confusion matrix will provide a detailed breakdown of the model's classification results by showing the number of true positives, true negatives, false positives, and false negatives.

The F1-score, on the other hand, will provide a balanced measure of the model's precision and recall, taking into account both the true positives and false positives. Together, these metrics will provide a complete picture of the model's performance, including its ability to correctly identify and classify different Filipino dishes.

Chart, treemap chart

Description automatically generated

*Figure 7.1 Confusion Matrix*

The confusion matrix in Figure 7.1 illustrates the performance of the model in terms of its ability to correctly classify different types of dishes. The columns represent the predicted classifications made by the model, while the rows represent the actual classifications. The matrix shows that the model struggled to accurately classify dishes, with only 220 out of 575 test cases being correctly identified, resulting in an accuracy rate of 38.26%. In particular, the model frequently misclassified Adobo as Sinigang and Pinakbet, and Pinakbet as Sinigang and Adobo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 2: Classification Report | | | | |
|  | Precision | Recall | F1-score | Support |
| Adobo | 0.41 | 0.42 | 0.42 | 197 |
| Pinakbet | 0.34 | 0.26 | 0.30 | 182 |
| Sinigang | 0.39 | 0.45 | 0.42 | 196 |
|  |  |  |  |  |
| Accuracy |  |  | 0.38 | 575 |
| Macro Avg. | 0.38 | 0.38 | 0.38 | 575 |
| Weighted Avg. | 0.38 | 0.38 | 0.38 | 575 |

As we can observe from table 2, the overall performance of the model on the Filipino Dish is not high. While the training and validation accuracy was able to reach the 75% given benchmark, the F1-Score does not.

There could be several reasons why the model performed poorly, some of which include: lack of quality data or the training data does not represent the actual data; and lack of diversity in training data.

In the following sections the researchers decided to train the model in fewer class, namely Adobo and Sinigang, to test the model on fewer and high scoring class.  
  
Text

Description automatically generated

*Figure 8.1 Last Three Epochs*

The validation accuracy and loss consisting of the new classes is significantly higher and lower than the previous validation accuracy and loss.

Chart, line chart

Description automatically generated  
*Figure 8.2.a Accuracy*

Chart, line chart

Description automatically generated

*Figure 8.2.a Loss*

Chart, treemap chart

Description automatically generated

*Figure 8.3. Confusion Matrix*

1. CONCLUSION

The results of the classification report indicate that the model performed poorly, with a low accuracy of 0.38 and low precision, recall, and F1-score across all classes. The model struggled to accurately classify dishes, frequently misclassifying Adobo as Sinigang and Pinakbet, and Pinakbet as Sinigang and Adobo. There could be several reasons for this poor performance, such as lack of quality data, lack of diversity in the training data, overfitting, insufficient data, lack of feature engineering, incorrect model architecture, hyperparameter tuning, unbalanced class distribution, noise in the data or human error. To improve the model's performance, it is important to investigate the cause of poor performance and take appropriate measures to address it. This may include collecting more and diverse data, carefully selecting and engineering features, experimenting with different model architectures, and tuning the hyperparameters.

The researchers used the model to identify fewer classes, namely Adobo and Sinigang. The training and validation achieved 98% accuracy, while also keeping the training and validation loss at around 10%. Furthermore, from the result that was given, the model has an acceptable performance. The accuracy of the model is 0.56, meaning that 56% of the dishes were correctly classified by the model. Overall, the model has an acceptable performance on fewer class and it is able to correctly classify dishes with a moderate rate.