

Tutorial II: Image Classification


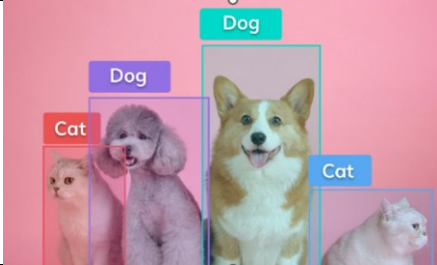
Acknowledgment

1. This tutorial was prepared by Pengxiang Jiang (pj016@uark.edu) at the University of Arkansas.
2. The dataset and algorithms used in this tutorial are from C. Dunlap, H. Pandey, and H. Hu, "Supervised and Unsupervised Learning Models for Detection of Critical Heat Flux During Pool Boiling," in *Proceedings of the ASME 2022 heat Transfer Summer Conference*, HT2022-85582. [\[link\]](#)
3. This tutorial was implemented in MEEG tech elective "Machine Learning for Mechanical Engineers." [\[link\]](#)

- Background of Image Classification:

Image classification is a type of specific machine learning application that falls within broad field of computer vision. Computer vision is a field of artificial intelligence study that focuses on training computers to understand, analyze, and extract meaningful information from various types of visual data, for example, videos or images. The ultimate outcome of these type of tasks is to replicate human vision by utilizing computer power, hence the name "Computer Vision".

There are many types of computer vision applications, the most popular ones include: image classification, object detection, image segmentation, object tracking, visualization generation, etc. Table 1. is a visual explanation of the tasks. For this modeling task, we will focus on image classification.

| Question (Types) | Input | Outcome |
|---|--|--|
| Is the object in the picture cat? (Image Cl) |  | Yes (Yes or No, binary answer) |
| What objects are in the picture and how many of them? (Object Detection) |  | Two cats and two dogs. (Detect object types and quantities) |

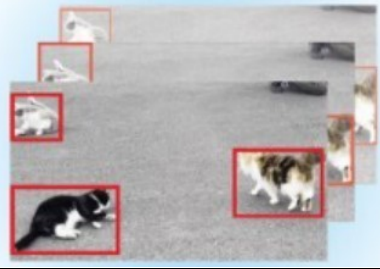
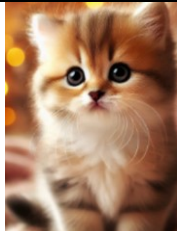
| | | |
|--|---|---|
| Where are the objects in the videos or multiple frames of pictures, and what is the moving direction? (Object Tracking) |  | One move to picture left, and one stay in the same place. (Track movement based across a sequence of frames) |
| Generate a picture. (visualization generation) | A cat, young, yellow fur, look up |  |

Table 1. Visual explanation of different types of computer vision task.

- **General code structure workflow & Convolutional Neural Network (CNN) Concept**

For this modeling task, we will focus on image classification. Similar to other python machine learning model, the main steps will include:

1. **Import library,**
2. **Import Data**
3. **Process the data (Data augmentation, Resizing, Splitting for Train-Validation-Test)**
4. **Build CNN layers**
5. **Flatten CNN outcome**
6. **Building Dense layers**
7. **Compile and Train the model**
8. **Test the model**
9. **Visualize the outcome.**

Most of the steps should already be introduced in homework 1, but there are some new concepts that need to be introduced here: Data Augmentation, CNN Network, CNN outcome flattening.

Data Augmentation: A technique used to artificially increase the size and diversity of a training dataset by applying various transformations to the existing data to create new training samples. Transformation types include, but not limited to, rotate, flip, resize, color alternate the images. Why? – Diversify the training database, so model can be trained on a more robust database to prevent overfitting and under training. Some readily available

python package: Imagegenerator from keras, torchvision from pytorch, albumentations, and imgaug.augmenters.

CNN Network: Convolutional Neural Network is a network designed primarily for processing structured grid-like data, such as images. It can extract “features” and learning spatial hierarchy from an image. CNN uses a user definable filter to scan through a image, and the filter will read spatial information and special pixel information from the scanned area, all these extracted information is know as “Feature”. Then a feature map will be generated to represent the original image. Throughout the training process, many feature maps will be generated from different image data.

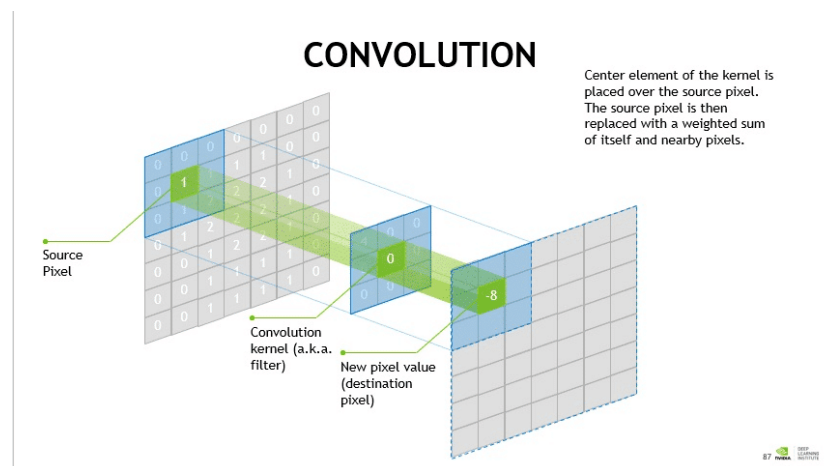


Figure A1. How filter extracts features.(Image credit: Nvidia)

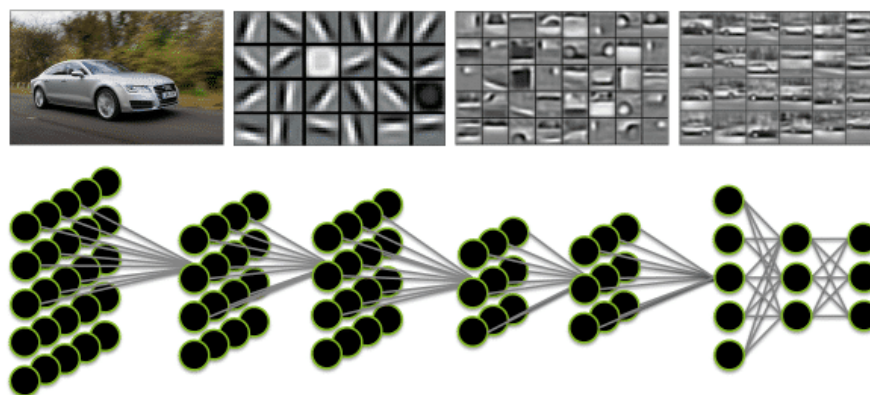


Figure A2. What CNN layers see (Image credit: Nvidia)

It may see “a diagonal line at Filter 1, a dark circle at filter 2, a white color square at filter 3, and etc” all these features will contribute to a whole feature map.

CNN outcome flattening: All the output feature maps from CNN network are multi-dimensional tensors, you can think of them as Image width x Image Height x Image color.

But input of downstream dense layer has to be 1 D array. Then you will need to flatten the multi-dimensional input into a 1 D array for dense layer to understand.

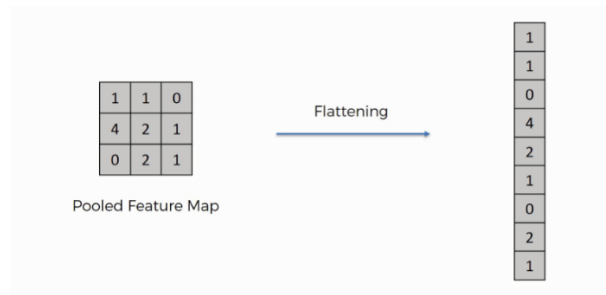


Figure A3. Feature Map Flattening

So, to sum up, following Figure A4 shows how CNN network works:

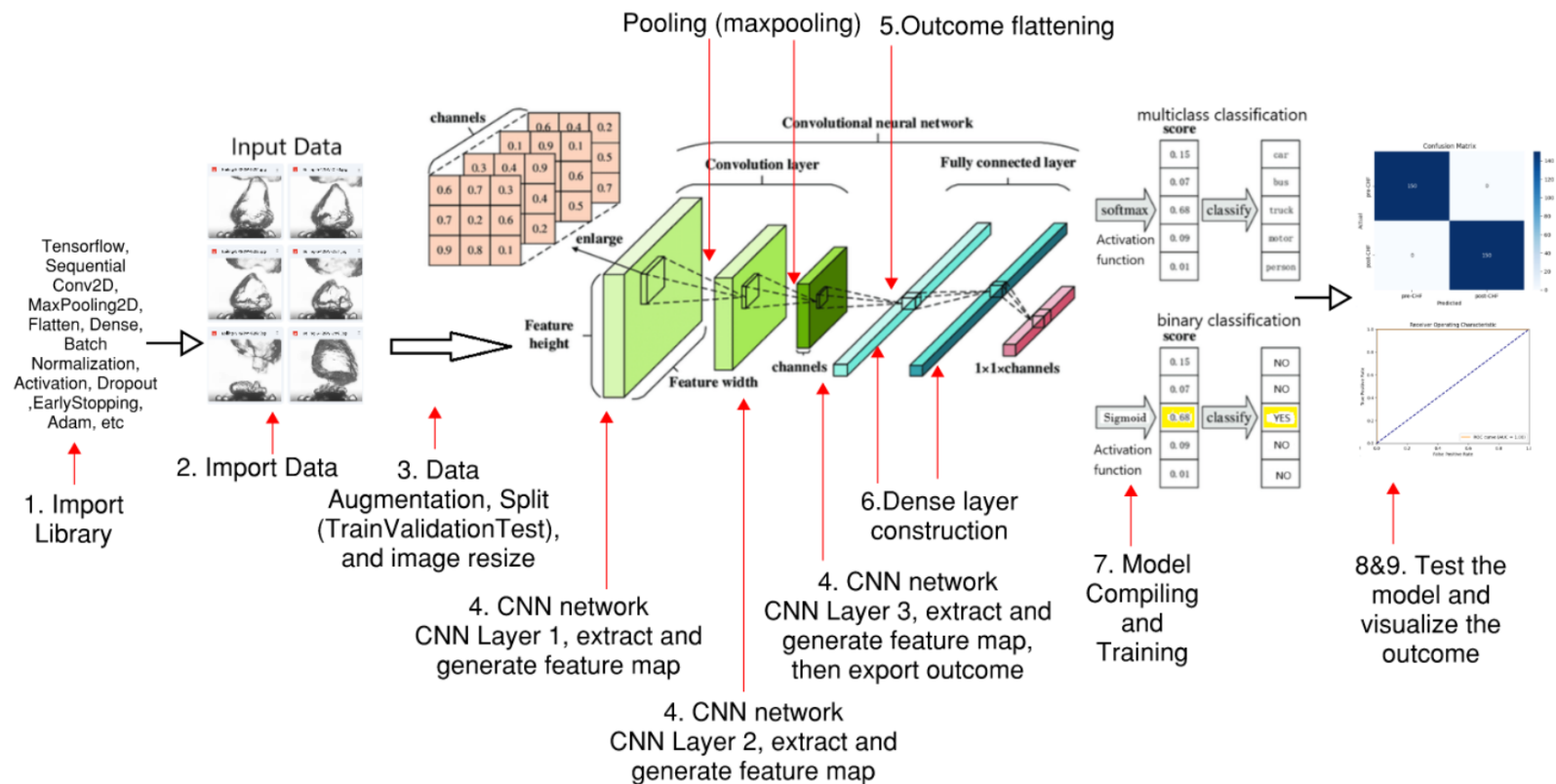


Figure A4: Image Classification Model Workflow

Now, we can proceed to tackle the task in Homework 2.

Homework 2 Task: **POOL BOILING CHF IMAGE CLASSIFICATION**

Section 1: Problem identification:

In pool boiling experiments and research, critical heat flux (CHF) is an important phenomenon that the maximum heat flux transfer at which a phase transition (from liquid to vapor) occurs on a heated surface before a significant change in the boiling regime. Once CHF is reached, the boiling process becomes unstable, leading to a sharp drop in heat transfer efficiency, can cause equipment failure. So, identify when the phenomenon happens is significant, one proposed method is to use bulb images taken from boiling process to predict or identify the phase of boiling process. For this project effort, we will build a machine learning model to help classify the pre and post CHF image.

Section 2: Solution description:

1. Basic methodology of the solution:

Because the bubble shape and formation has distinct features before and post critical heat flux point, so we can use an image classification machine learning model to read and classify the picture based on their features. As described above in the background section. We will use CNN layers to extract features from image dataset and flatten export to dense layer for training. The detailed steps and code as described below.

2. Code explanation and execution

The code is written in python language, version 3.10; development environment is Google Collaboration Laboratory (Colab), all the tools and libraries are pre-built into the integrated development environment (IDE).

The creation of this code has consulted generative AI ChatGPT, also partially generated by Gemini AI that integrated into Colab IDE.

Line by line explanation and code structure is explained in the notebook. All the steps are correlated between python notebook code and explanation below, you can use information below as commentary of the original code.

Steps 0. Install package that not presinstalled in Colab environment.

The split-folder package will physically generate new folders with split image data inside and label the folder as you indicated. This split method help me read and

understand the splitting process, because I can physically visualize the split images. However, it has drawback, you will need to manually delete or add code to delete the split folder, before you rerun the entire code again. Otherwise, you will end up have more added split folders.

Steps 1. Import Libraries

Data splitting tools: “splitfolders” as described above.

Image reading and processing tool: “ImageDataGenerator”

ImageDataGenerator (IDG) is used for image dataset import and recondition. This tool is integrated into Keras, and it can load image file batch by batch on the go, this technique can help reducing memories usage during modeling, and it is specially build for CNN type model, which is this model based on; it can further improve computation time. However, it is more for deep learning, and has less manual option to recondition image, but I want to make this model more general and can be adapted to other more complicated problem; so this pack is the ultimate choice. One of the biggest benefit of ImageDataGenerator is that it has integrated augmentation function, it offers image manipulation ability like rotation, offset and mirror the image files to create more generalized model. However, during model debugging process, this function actually cause severe overfitting, so it is disabled. But can be re-enable by removing the “#” in front the augmentation lines.

There are other tools you can use, for example: OpenCV (cv2) or PIL? They are not memory efficient as IDG, and this is more suitable for k-NN or SVM model, load small dataset. Even though the dataset of this modeling effort is small, but IDG has wider application range. The notebook code will give example of how to using cv2 package as comparison.

Model building tool: “tensorflow” (models, layers, callbacks, optimizers): tensorflow is one of the most widely used main machine learning frameworks, it contains almost all the tools that needed for any types of machine learning model construction.

Performance visualization tool: “sklearn” package. This includes confusion matrix, roc_curves and etc

Plotting tool: “matplotlib, seaborn, and etc.”

Steps 2. Data Import

I saved all my image files on google drive for quick access, so data import will start from mount google drive. You may start your data import from your local drive. Example code of importing from your local drive will be given in the notebook.

Steps 3. Data Processing

a.

As mentioned above, all the image data will be split into test-validation-training, three distinct portions; and split folder tool will be used, therefore, the output folder directory will be redefined.

An example code of using sklearn train-test-split will be given as comparison.

b.

IDG has integrated augmentation abilities. Since, this modeling task is simple, over augmentation will like to cause overfitting; so the code is disable, you may try and see if it works for you. A different code example of using cv2 package will be given as comparison.

c.

This section will define the split and process image directory.

Steps 4. Build CNN layers

The model will be set up with convolution layers – dropout layers – dense layers – dropout layers – output layers format. Two extra dropout layers are added to increase the generalization of model and prevent overfitting. Each of the convolution layers will be provided with data normalization for better stability and faster performance, and maxpooling to down sampling the image, down sampling filter is set to 2x2, since our image files have been resized to very small to begin with, using 2x2 will not be any issues. Total of 3 convolution layers have been utilized for better model generalization.

There are some notable item need to be explained and pay attention:

ReLU (Rectified Linear Unit) Activation Function: The ReLU activation function is a popular choice in deep learning models due to its simplicity and effectiveness. It introduces non-linearity to the network, enabling it to learn complex patterns. However, it has its problem, AKA "**dying ReLU**" problem, where neurons can become inactive if their inputs are consistently negative. This can lead to a portion of the network not learning. You can try "Leaky Relu", "ELU", or "PReLU" if above issue happens to you.

Maxpooling: It is a pool layers that reduce the spatial dimensions and extract features from input images. The size of it can impact both accuracy and training speed. A “too large” maxpooling parameters can cause loss of information, reduced resolution, and over-simplification. A “too small” maxpooling parameters can cause inefficient downsampling, slow training time, and redundant feature extraction. So, if you experience large modeling training errors or slow training time, you can try to adjust maxpooling parameters.

If we use pre-trained readily available CNN model, the entire process will be super easy. A VGG16 model code will be given in the notebook for comparison.

Steps 5. Flatten CNN outcome

This step is very straightforward and explained previously.

Steps 6. Build Dense Layer

This is a common artificial neural network layer. But there is one concept that need explanation: Sigmoid.

The activation function of image classification has two common forms: sigmoid and softmax. Sigmoid only differentiate value below or above certain number, larger than the threshold will be considered as 1, smaller than the threshold will be considered as zero. So it is good for binary classification (This task). However, Softmax can record range of the value, so it is good for multi-category classification.

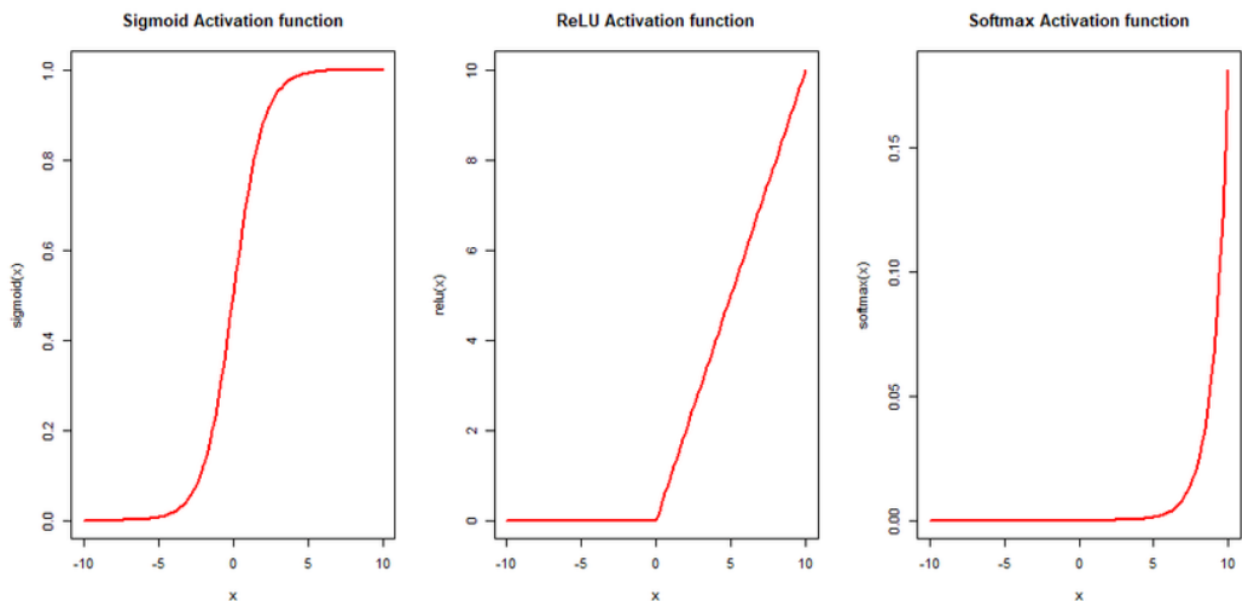


Figure B1. Sigmoid VS Relu VS Softmax

Steps 7. Compile the model

This step compiles all the above layers together and execute the training. There is also two important hyperparameters worth mentioning: learning rate and epochs

Learning rate: Controls how much the model weights are adjusted during training. Too high: Faster training but risk of overshooting the optimal weights. Too low: Slower training but potentially better convergence to a global minimum.

Epochs: Number of complete passes through the training dataset. Too high: Model may overfit, especially without proper regularization. Too low: Model may underfit.

Steps 8. Evaluate the model performance.

Model will be evaluated by using the pre-split image data file from the very beginning. And loss function accuracy, validation accuracy, confusion matrix, AUC, and ROC will be plotted to visualize the robustness of this model.

Section 3. Result Interpretation

The figures below show the training accuracy and loss function errors per epoch. We can tell they both converge extremely fast, and reached 100% accuracy and 0% errors. Since this is a very simple task, these results are expected. However, for more complicated task, I may need to increase the epoch number.

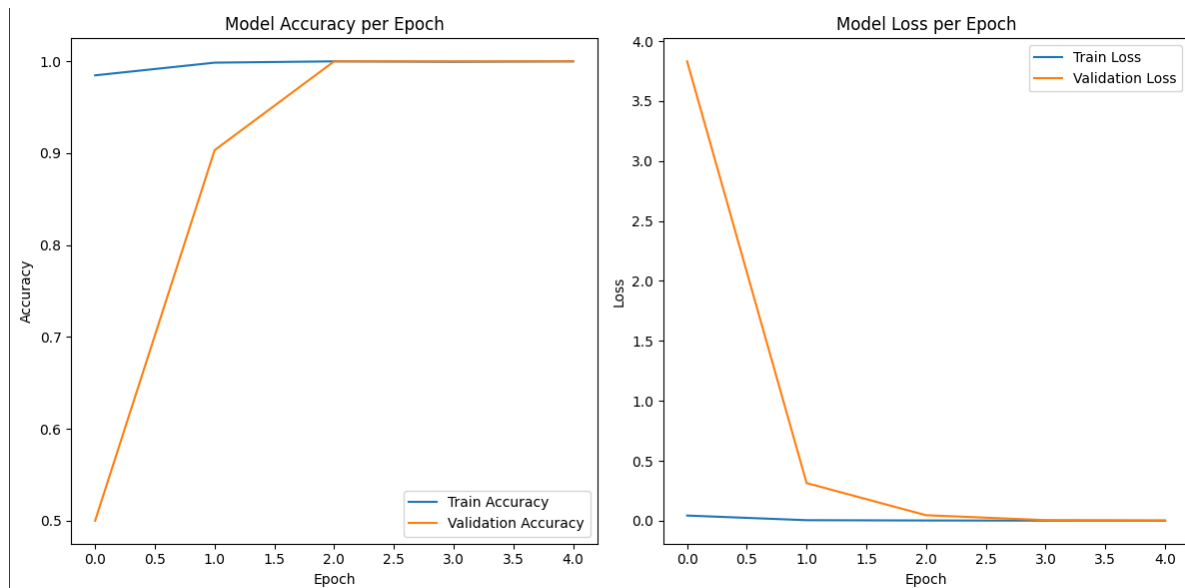


Figure C1: Modeling Accuracy Visualization

The confusion matrix indicates that the model is successfully trained with good accuracy.

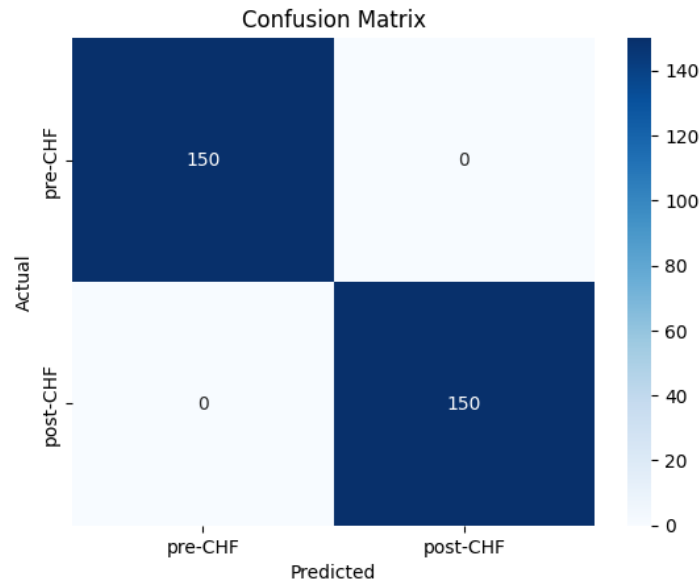


Figure C2: Confusion Matrix.

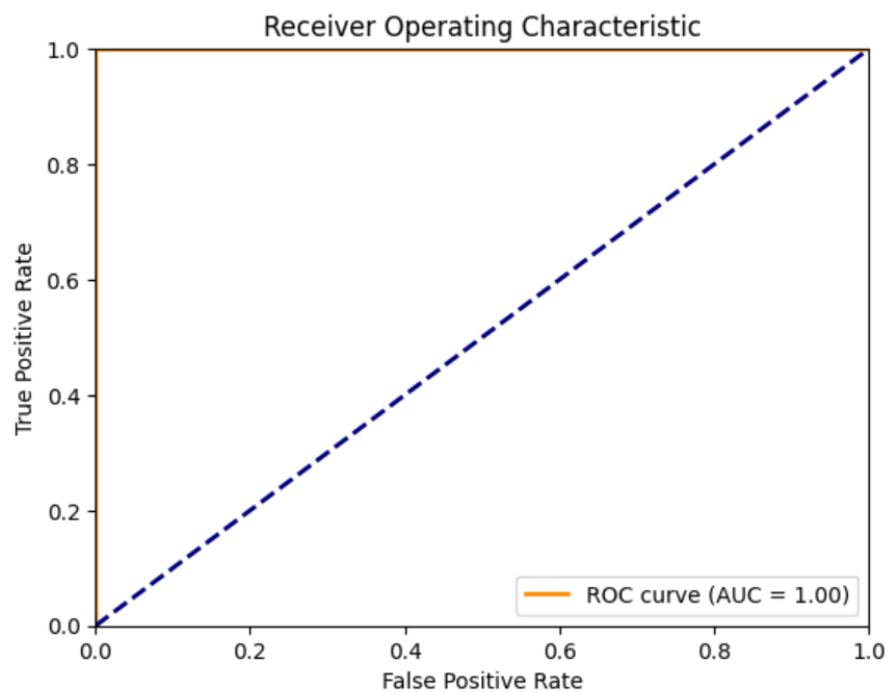


Figure C3: ROC curve.

Section 4. Some tips, Troubleshooting, and future improvements

3000 images are not small amount of data to upload, you might start your project early, just start to upload all the image, depending on the network traffic, it can take a while or

even several hours. Or sometime, Google drive may straight up freeze due to too many single files in the folder.

Overfitting happens a lot. Please tuning your hyperparameters for the best result. But, most of the time, it is due to model set up too complicated (over thinking), you may simplify the model setup.

If your confusion matrix look incorrect, for example, predict and actual value are close to half and half. You may need to make sure the confusion matrix data shuffle set to "False".

Future improvement for this model can be to reconstruct the code and try to add attention map (GradCam);also try with more complicated task and visualize how attention map works.