Image Recognition and Classification

Presented by Aung Lin Htet and
Wai Hlyan Win

Objective

Salmon Sashimi - Wai Hlyan Win

&

Egg Tart - Aung Lin Htet

Data collection

```
# DATA PATHS
train_dir = 'C:/Project/SSET/datasets/train'
valid_dir = 'C:/Project/SSET/datasets/valid'
test_dir = 'C:/Project/SSET/datasets/test'
```

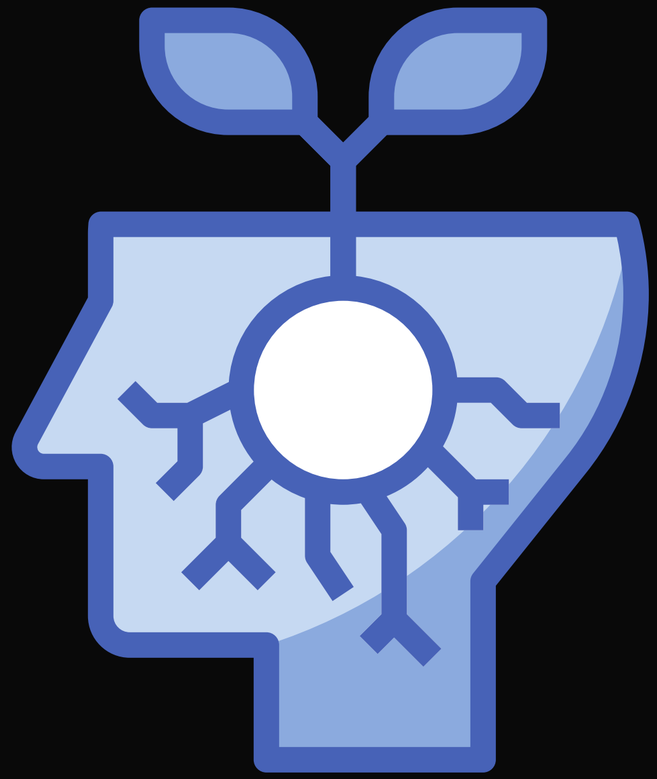
Salmon Sashimi - 186 , 40, 43 (269)

Egg Tart - 479, 108, 108 (695)

Unknown - 494, 119, 105 (718)

Model Preparation

We use Transfer Learning with
Inception V3 first
even though we are not allowed
to
we Just want to learn from it and
only when that works then we try
to test the model that we start
from scratch with trial
and error method. and try to
show it to a lot of variations by
being simple.



Data Splitting

- Data are split manually 70% Training, and 15 % each for testing and validation
- To have more control over variety and balance between the data set
- To reduce randomness and ensure that each set had a fair distribution of images

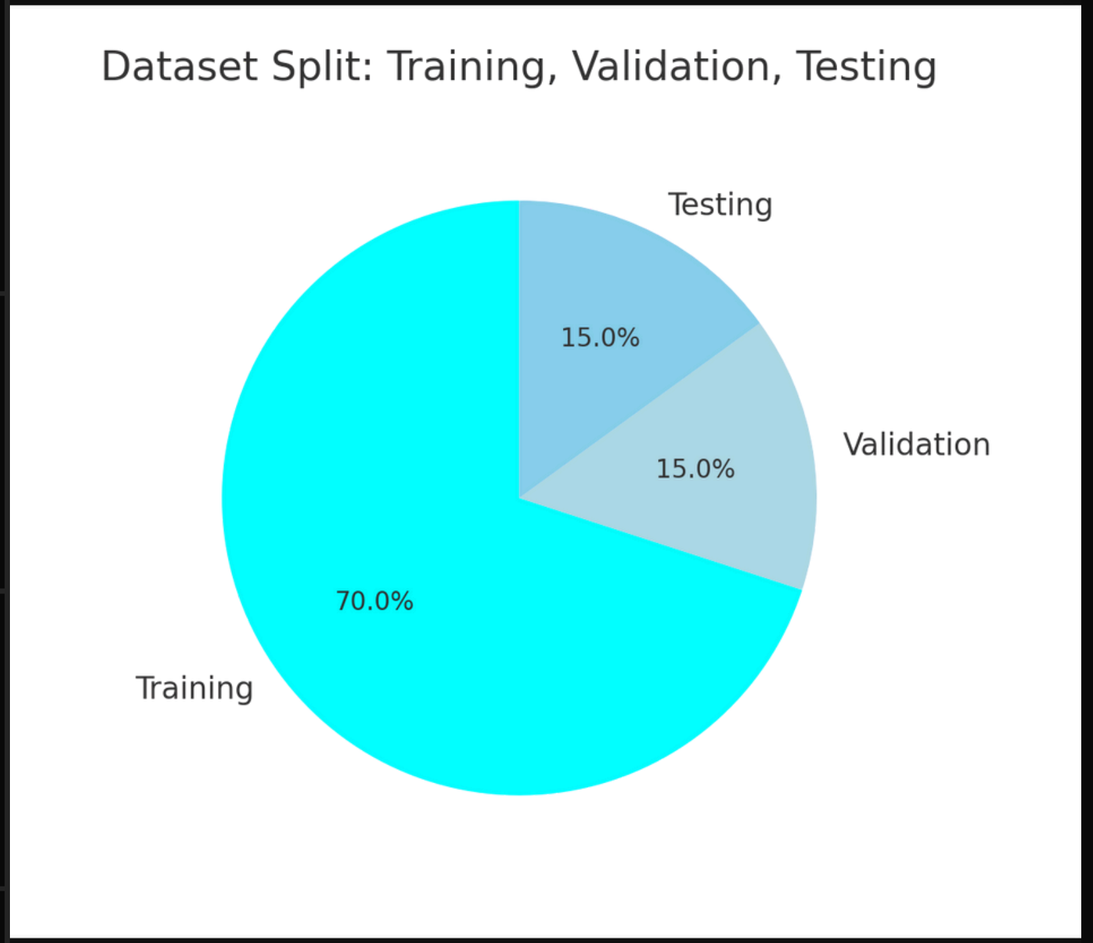


Image Preprocessing

```
# AUGMENTED TRAIN GENERATOR (for generalization)
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# VALIDATION & TEST GENERATORS (no augmentation)
valid_datagen = ImageDataGenerator(rescale=1./255)
test_datagen  = ImageDataGenerator(rescale=1./255)
```

All pixel values are converted from integers (0–255) to floats (0.0–1.0) for faster convergence during training.

- **Generate Image data from training set with different augmentation**
- **This will help the model to learn from different variety**
- **Test and validation data are not augmented to have a fair evaluation of the model**

Model Building

```
# MODEL DEFINITION
model = Sequential([
    Input(shape=(75, 75, 3)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(3, activation='softmax')  # 3 classes
])
```

- Create a model instance of sequential class
- Add convolutional layers to the model
- Flatten
- Add dropout layers to prevent overfitting and dense layers with decreasing units for gradual compression
- Finished with a Dense layer with 3 units and a softmax activation, suitable for classifying 3 categories: Egg Tart, Salmon Sashimi, and Unknown.

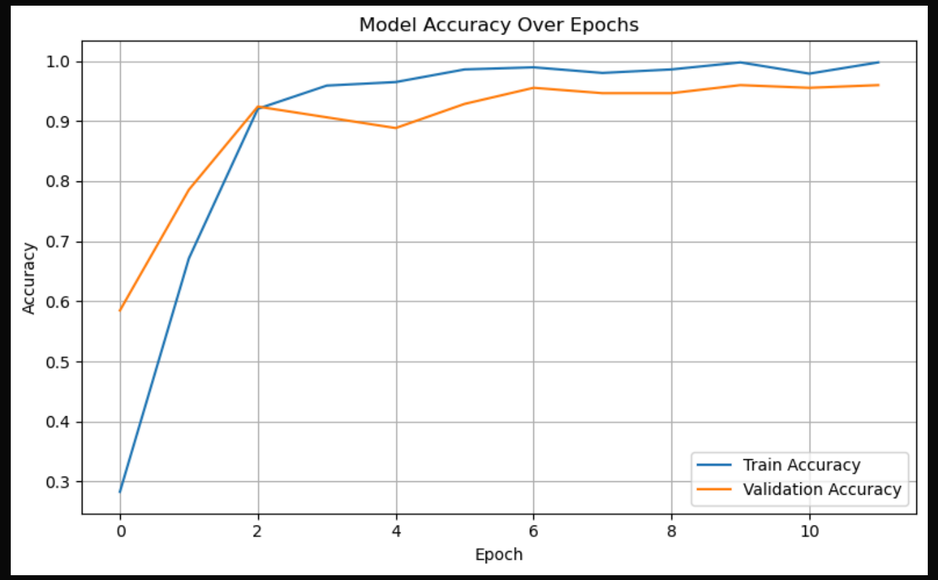
Model Training

```
# TRAINING
history = model.fit(
    train_generator,
    epochs=13,
    validation_data=valid_generator,
    class_weight=class_weight_dict
)
```

Visualization and Evaluation

```
# PLOTS - ACCURACY
plt.figure(figsize=(8, 5))
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

# PLOTS - LOSS
plt.figure(figsize=(8, 5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

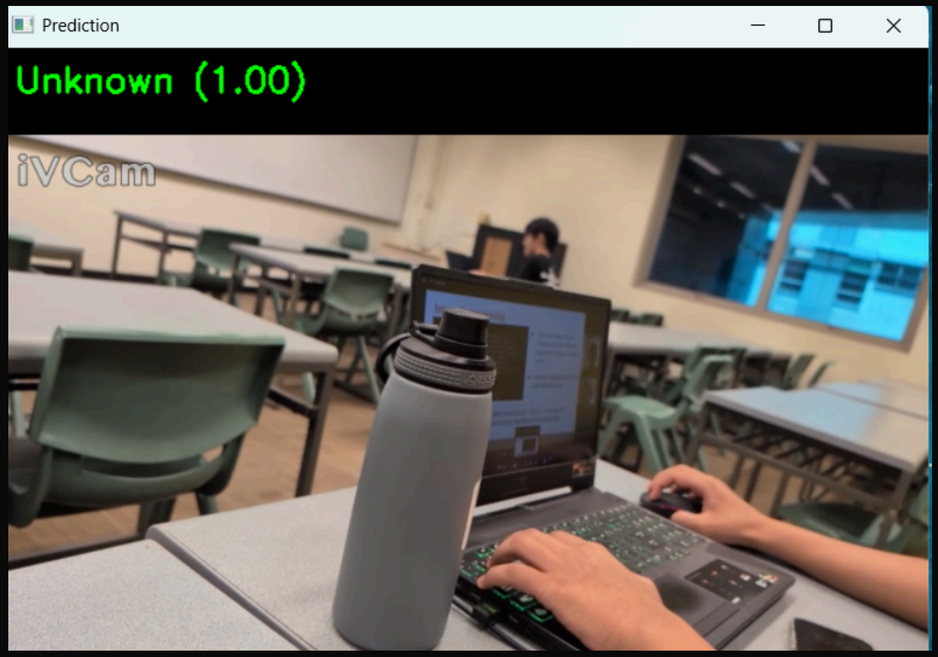
- Use matplotlib to plot the accuracy and loss changed over epoch
- Compare the results between training and validation



C:\Project\SSET\detect.py

detect.py | SSET_app.py | SSET_app_model.py

```python
import cv2
import tensorflow as tf
import numpy as np
from PIL import Image, ImageOps

model = tf.keras.models.load_model('sset_model.h5')
labels = ['Egg Tart', 'Salmon Sashimi', 'Unknown']

def predict_frame(image):
    image = ImageOps.fit(image, (75, 75), Image.LANCZOS).convert('RGB')
    image = np.asarray(image).astype(np.float32) / 255.0
    image = np.expand_dims(image, axis=0)
    prediction = model.predict(image)
    confidence = np.max(prediction)
    label_index = np.argmax(prediction)
    return f"{labels[label_index]} ({confidence:.2f})" if confidence >= 0.7 else "Unknown (Low confidenc

cap = cv2.VideoCapture(2)

    ret, frame = cap.read()
    if not ret:
        break
    cv2.imwrite('temp.jpg', frame)
    image = Image.open('temp.jpg')
    label = predict_frame(image)
    cv2.putText(frame, label, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)
    cv2.imshow("Prediction", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

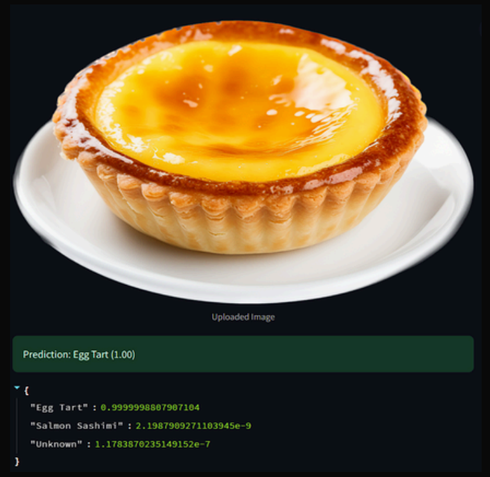


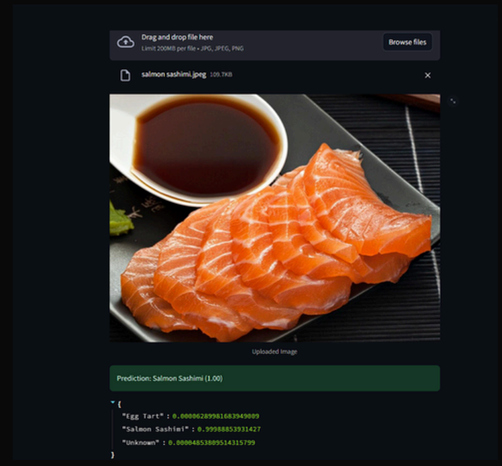
- Live detection using Android phone camera via (IVCam). The video feed is streamed over IP and processed in real-time using our trained model to identify Egg Tart, Salmon Sashimi, or Unknown.

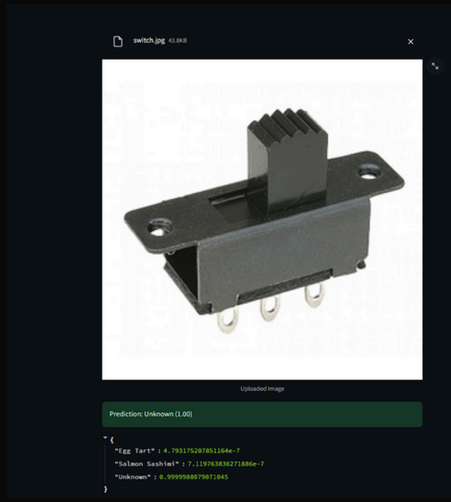
Image Testing

```python
import streamlit as st
import numpy as np
import tensorflow as tf
from PIL import Image, ImageOps

model = tf.keras.models.load_model('sset_model.h5')
labels = ['Egg Tart', 'Salmon Sashimi', 'Unknown']

def predict_image(image):
    image = ImageOps.fit(image, (75, 75), Image.LANCZOS)
    image = np.asarray(image).astype(np.float32) / 255.0
    image = np.expand_dims(image, axis=0)
    prediction = model.predict(image)
    confidence = np.max(prediction)
    label_index = np.argmax(prediction)

    if confidence < 0.7:
        return "Unknown (low confidence)", prediction[0]
    return f"{labels[label_index]} ({confidence:.2f})", prediction[0]

st.title(" Egg Tart vs Salmon Sashimi vs Unknown")
file = st.file_uploader("Upload an image", type=["jpg", "jpeg", "png"])

if file:
    img = Image.open(file)
    st.image(img, caption="Uploaded Image", use_column_width=True)
    label, probs = predict_image(img)
    st.success(f"Prediction: {label}")
    st.write({labels[i]: float(probs[i]) for i in range(3)})
```

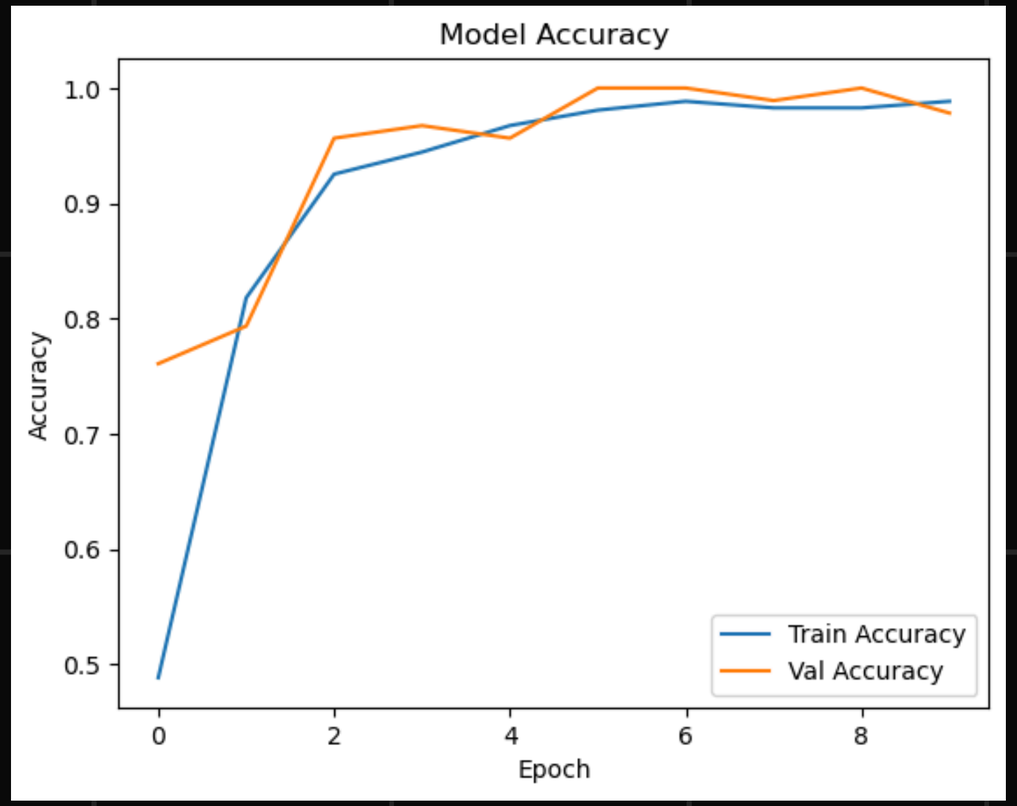
Webpage with Streamlit to test and see the prediction of the images

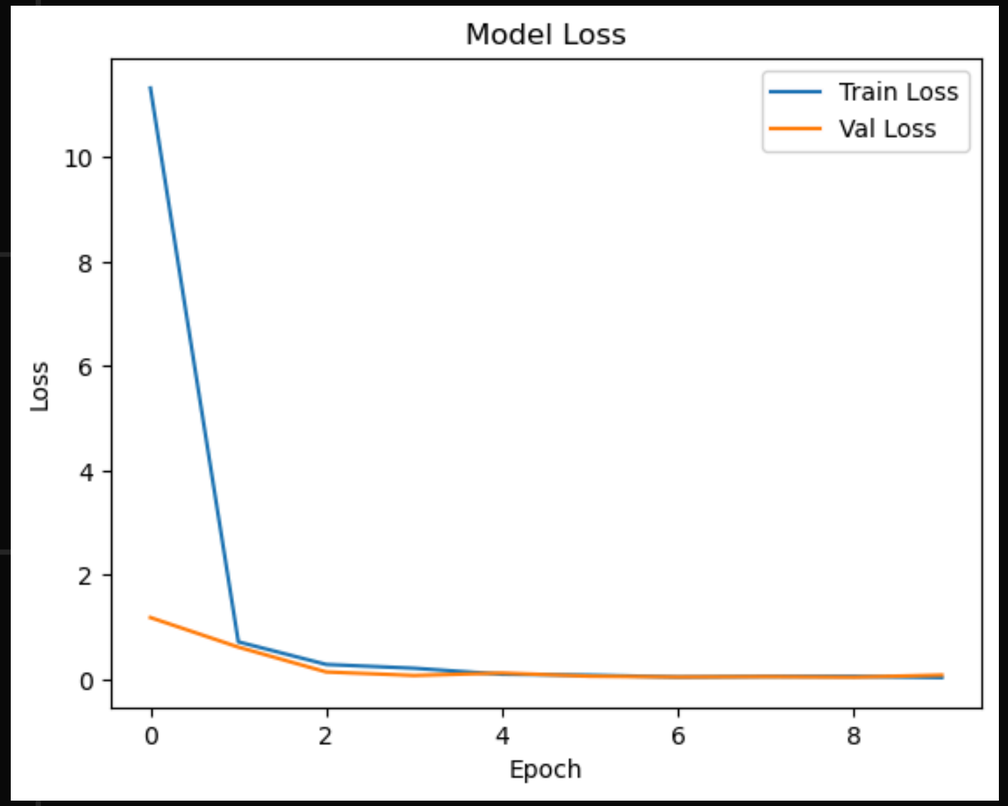






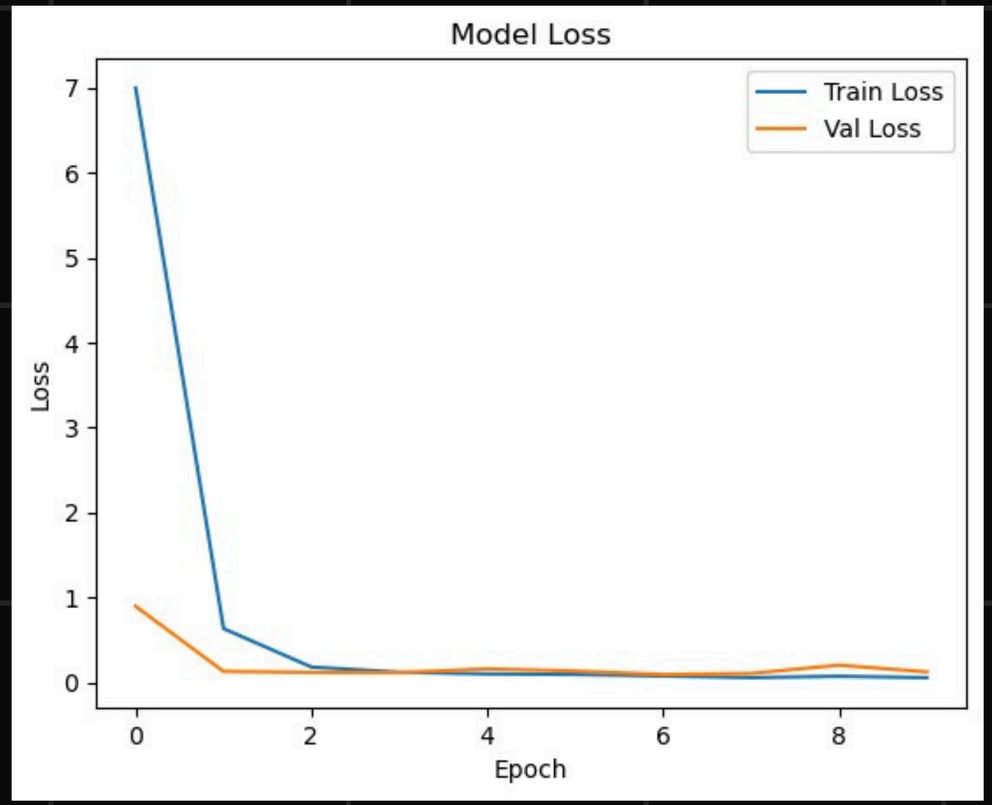
Model 1

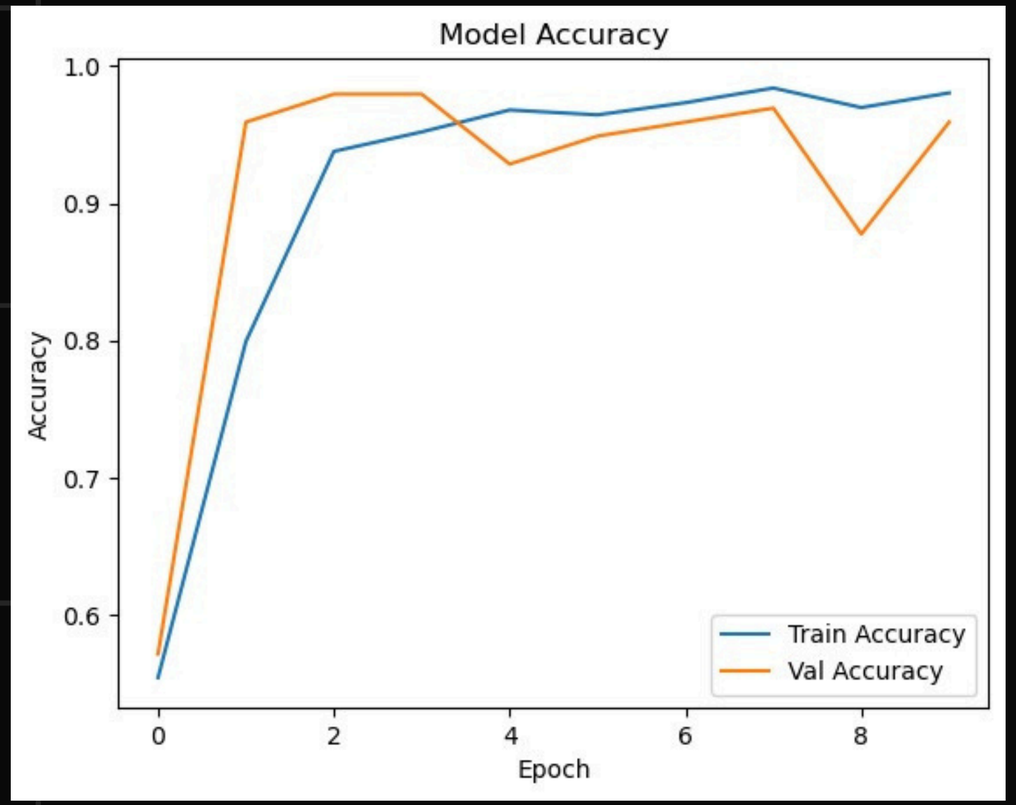




Used Inception V3 (Pre-trained) to show either Salmon Sashimi or Egg Tart

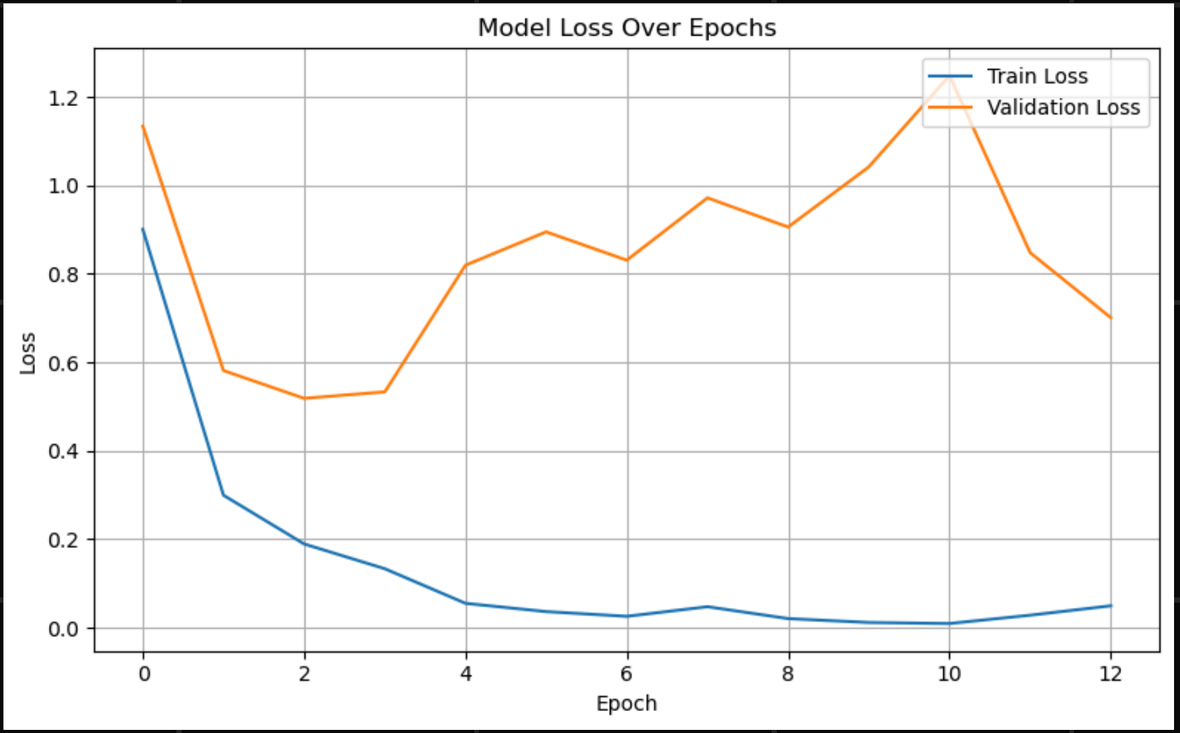
Model 4

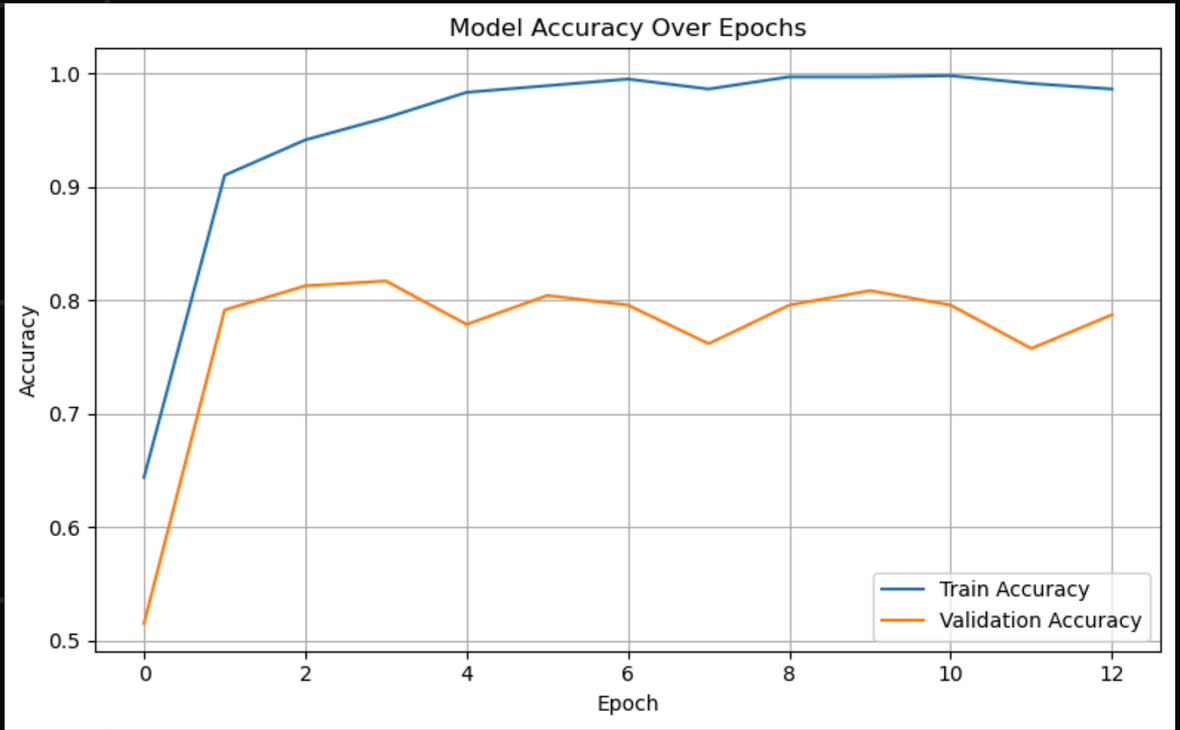




We started to build our model from scratch with only Salmon Sashimi and Egg Tart

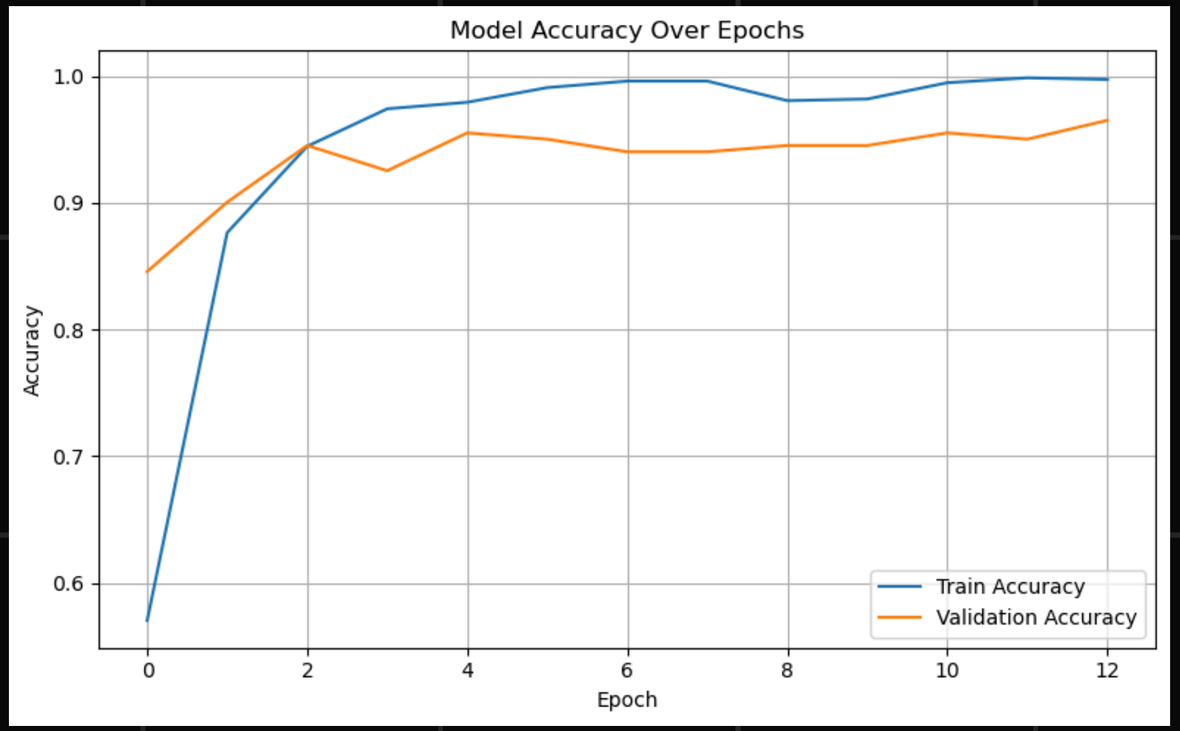
Model 15

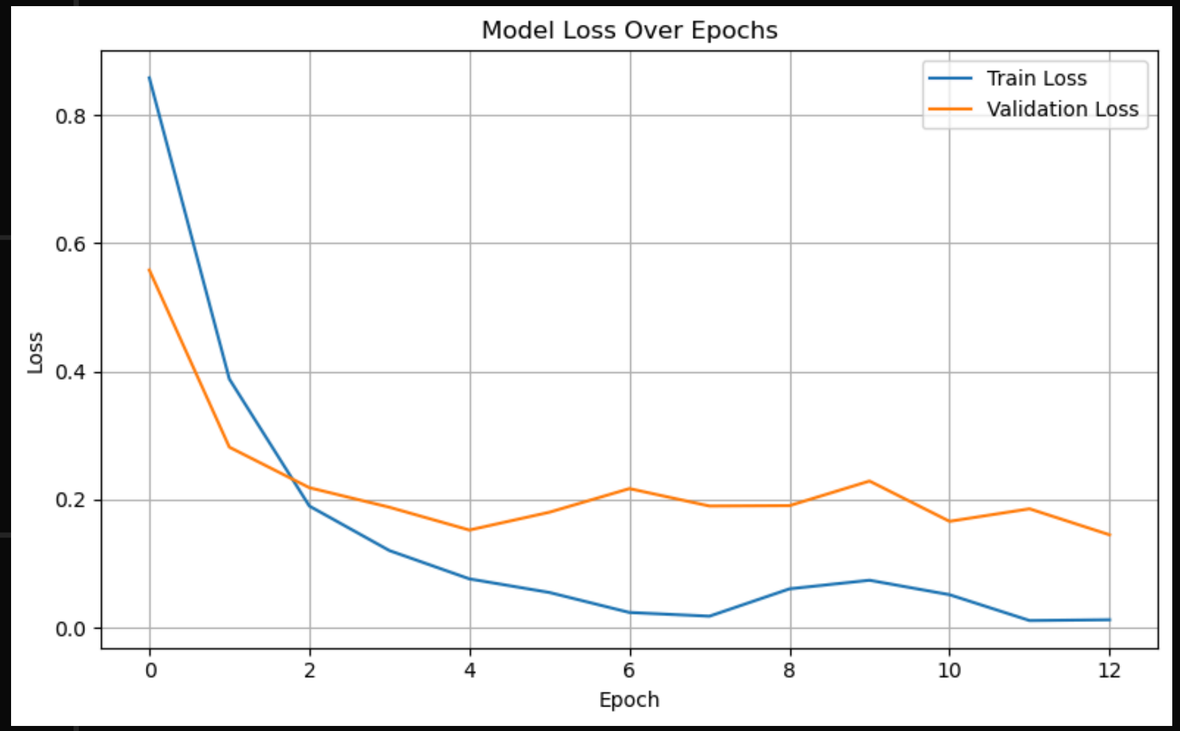




Problem started when we try to put unknown as a data class since everything is unknown

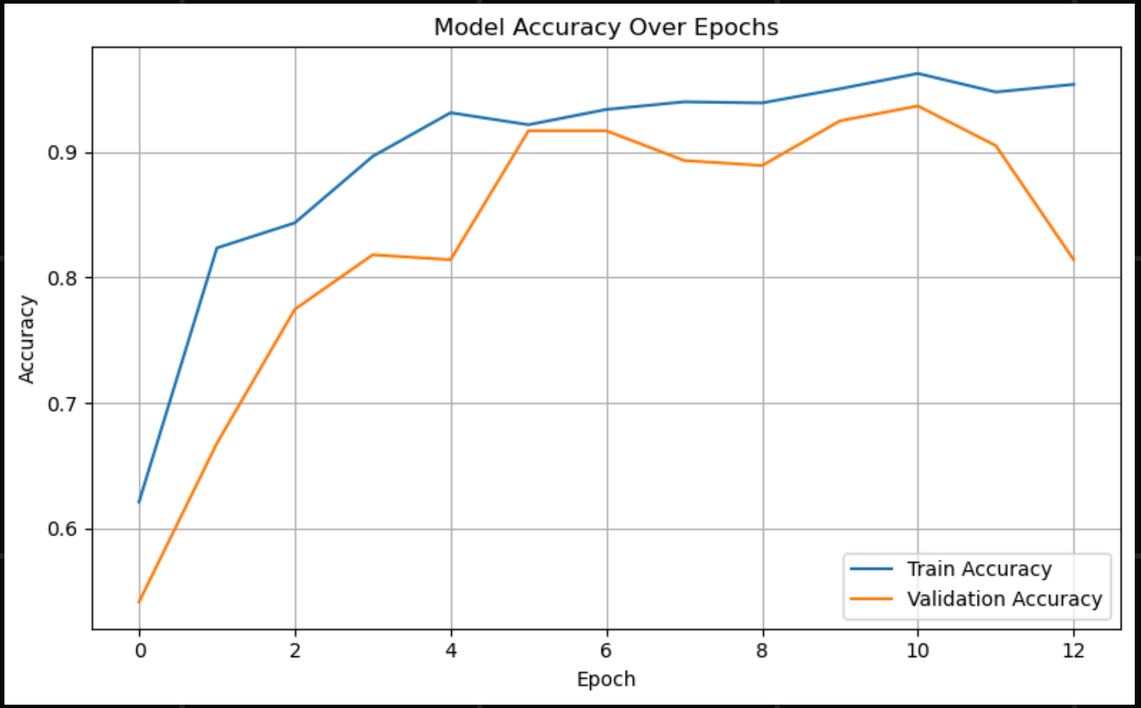
Model 18

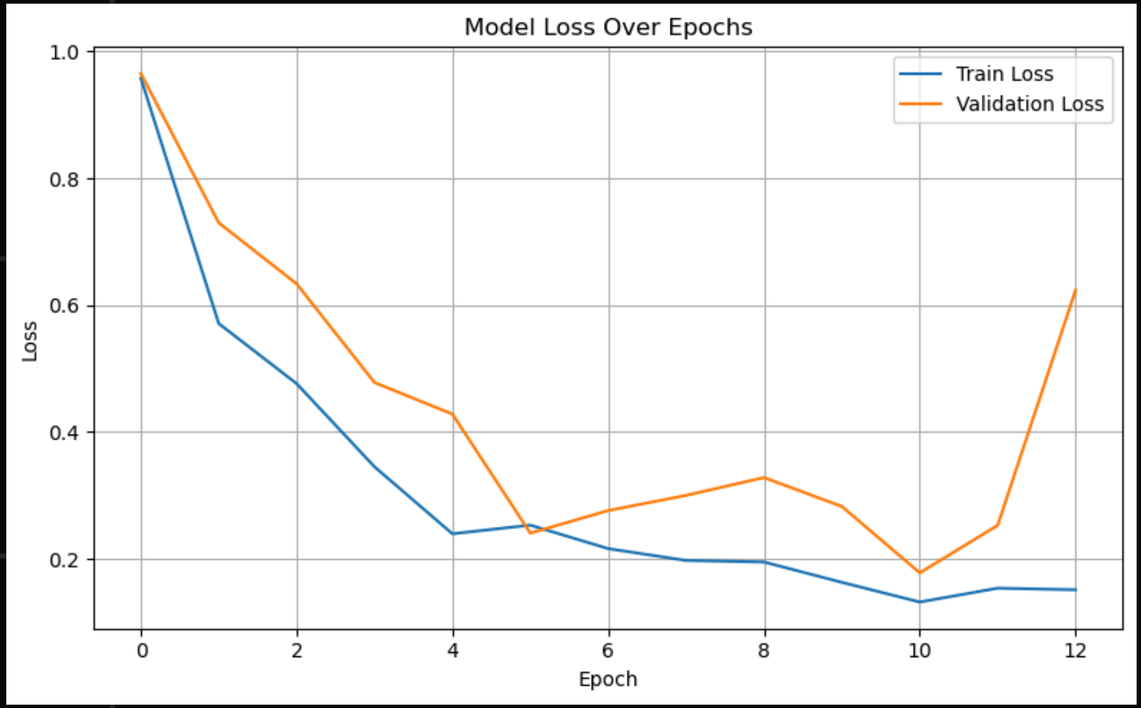




After cleaning up the data, we use only the self-source photo.

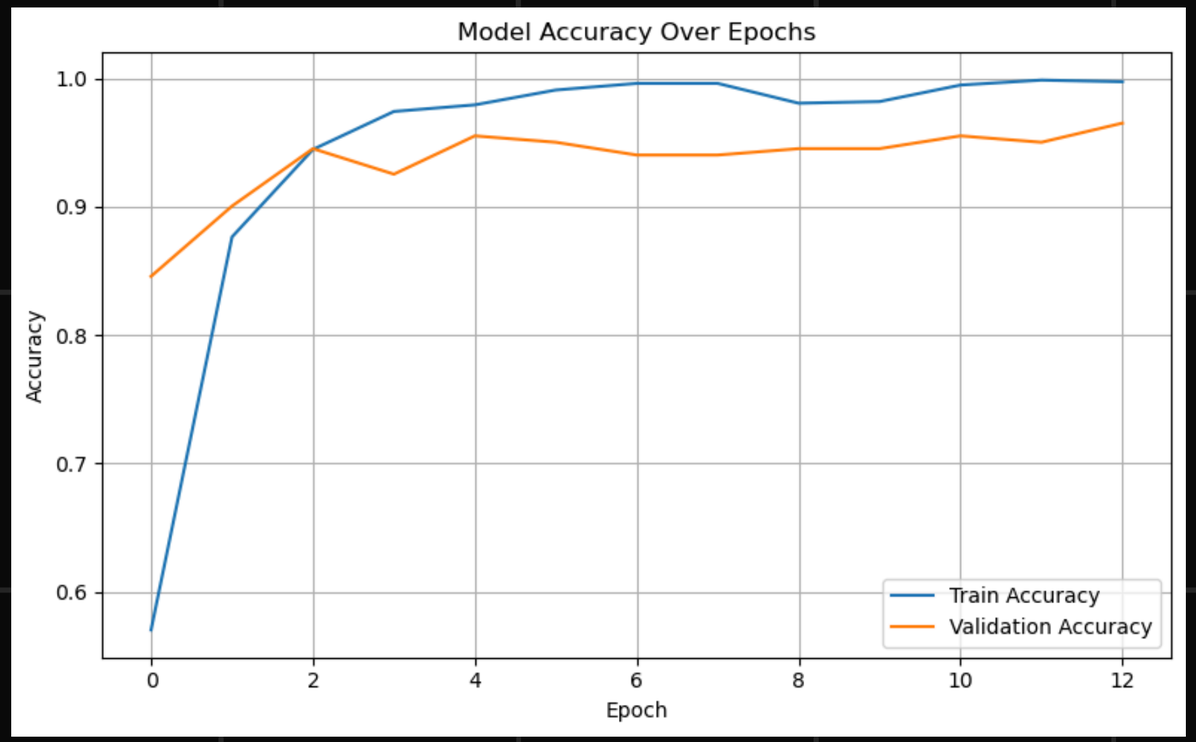
Model 20

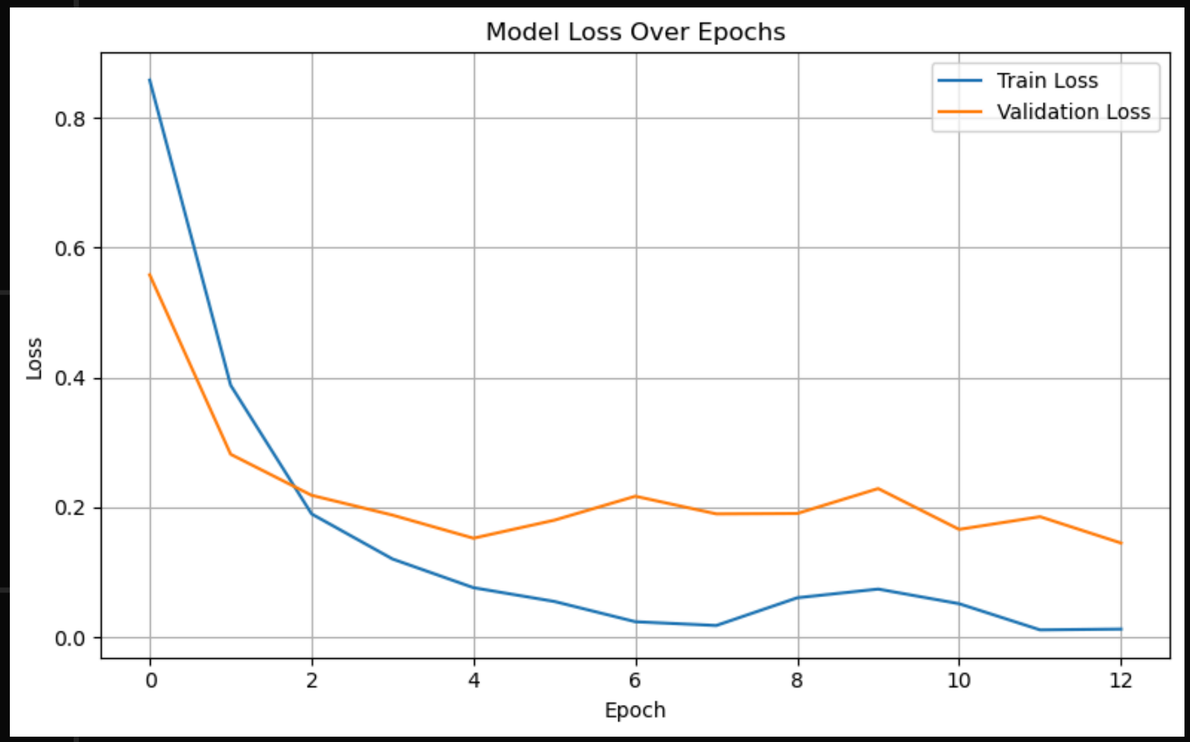




Result from using python code to auto split the data 70 , 15 , 15 for training , testing and validation respectively.

Model 25 (Best trained - model)





Our model now fully dependent on self-source data that we have taken even though the graph might not have been perfect but the result got better

Model Summary

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 73, 73, 32)        896
 max_pooling2d (MaxPooling2D)(None, 36, 36, 32)        0
 conv2d_1 (Conv2D)           (None, 34, 34, 64)        18496
 max_pooling2d_1 (MaxPooling2D)(None, 17, 17, 64)      0
 flatten (Flatten)           (None, 18496)             0
 dense (Dense)               (None, 128)               2367616
 dropout (Dropout)           (None, 128)               0
 dense_1 (Dense)             (None, 3)                 387
=================================================================
Total params: 2,387,395
Trainable params: 2,387,395
Non-trainable params: 0
```

- 3 Convolutional Layers - Each followed by a MaxPooling2D layer to downsample the feature maps.
- Batch Normalization - Added after the final pooling layer to stabilize and accelerate training.
- Flatten Layer - Converts the 3D output from convolutional layers into a 1D vector for dense layers.
- 3 Dense Layers at the End - Includes ReLU-activated layers and a final softmax layer with 3 units for classification.
- Dropout Layers - Placed between dense layers to reduce overfitting and improve generalization.

Test Results Using Streamlit Web Page Using Unseesn Footage from the Internet

Uploaded Image

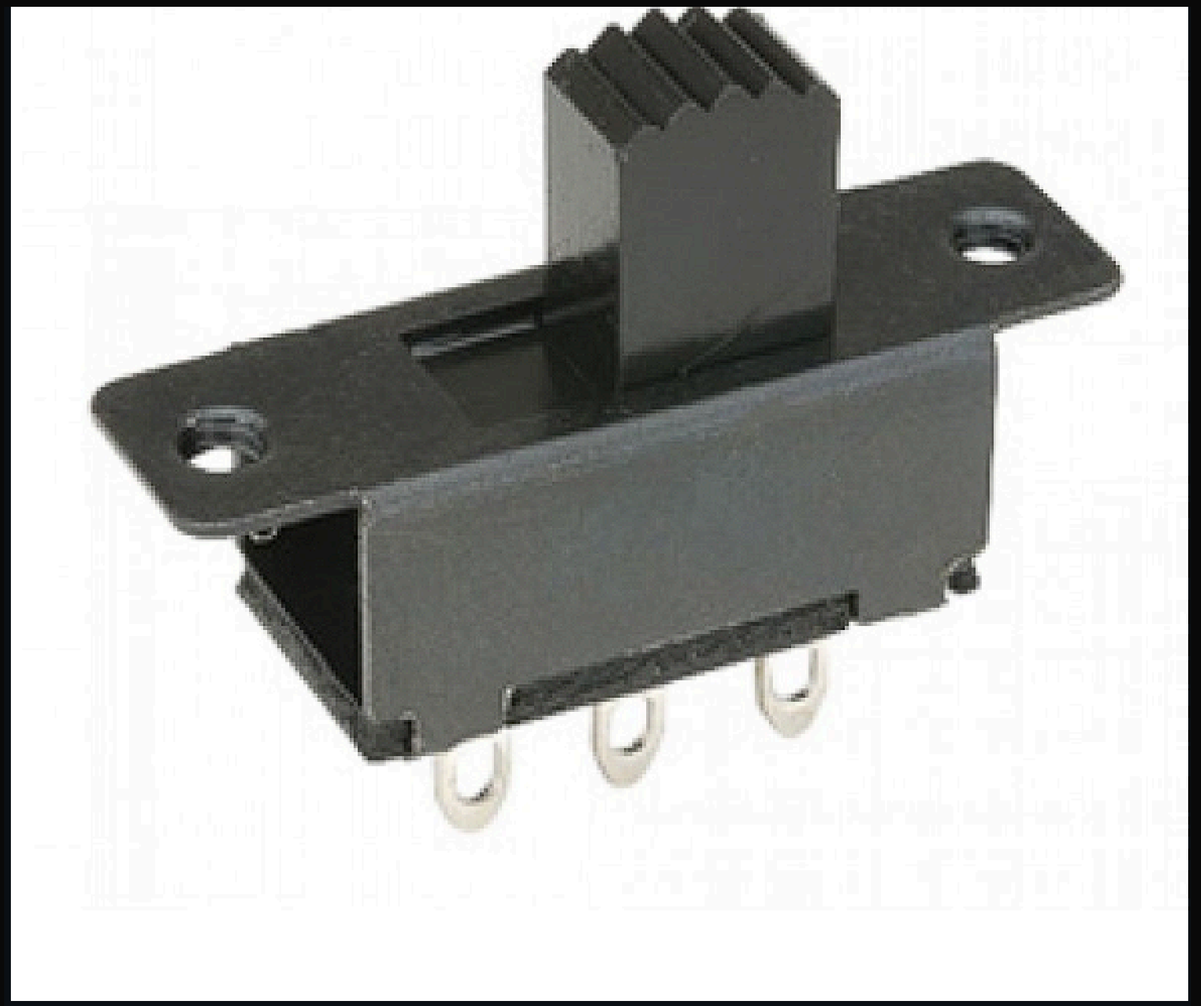
Prediction: Egg Tart (1.00)


```
{
  "Egg Tart" : 0.9999998807907104
  "Salmon Sashimi" : 2.1987909271103945e-9
  "Unknown" : 1.1783870235149152e-7
}
```

Uploaded Image

Prediction: Salmon Sashimi (1.00)


```
{
  "Egg Tart" : 0.00006289981683949009
  "Salmon Sashimi" : 0.99988853931427
  "Unknown" : 0.00004853809514315799
}
```

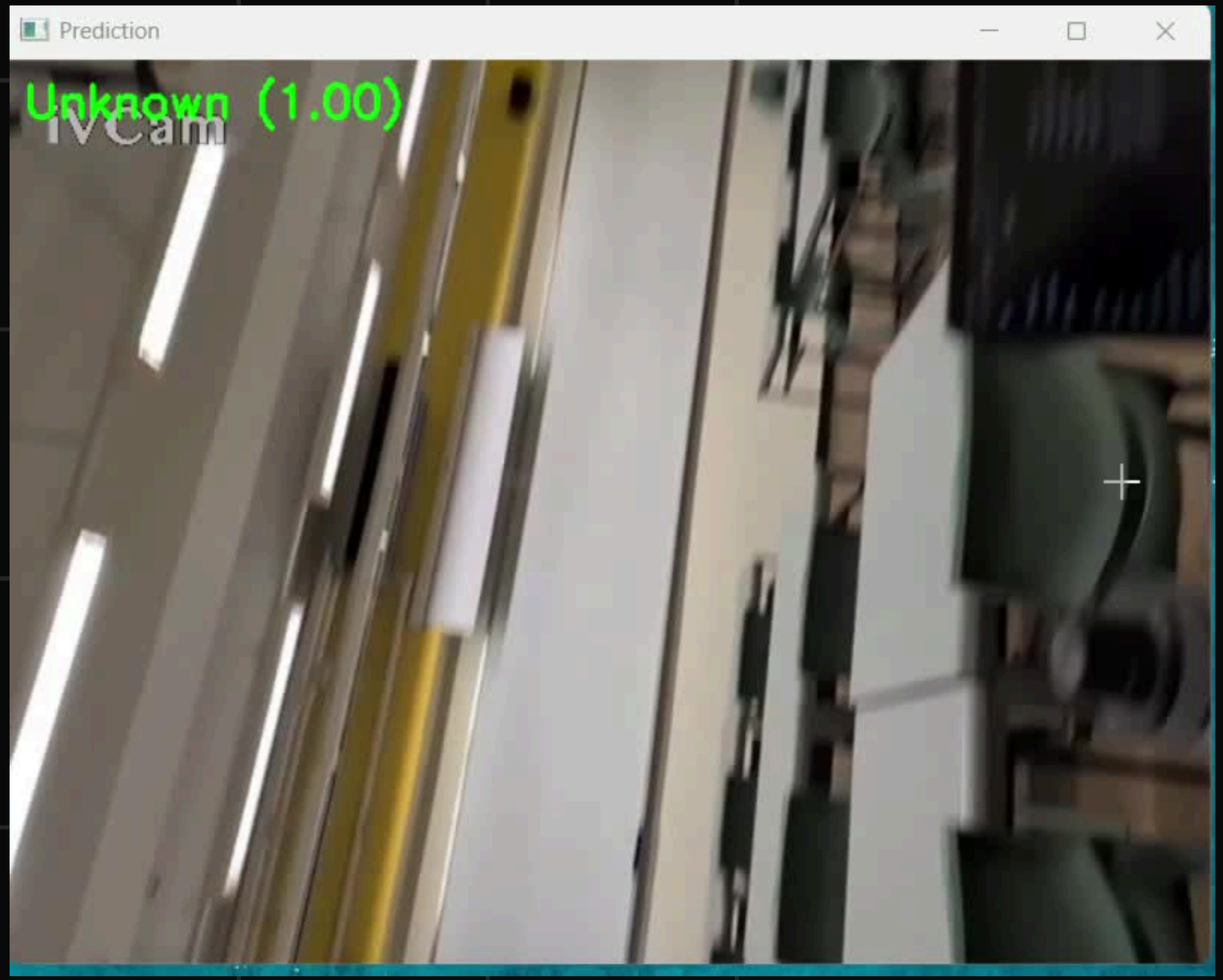


Uploaded Image

Prediction: Unknown (1.00)


```
{
  "Egg Tart" : 4.793175207851164e-7
  "Salmon Sashimi" : 7.119763836271886e-7
  "Unknown" : 0.9999988079071045
}
```

Live Detect Video



Problem and findings

- In AI model training, one of the biggest challenges is not just the model architecture, but the quality and consistency of the dataset.
- If the picture are too diverse, the model try to memorize rather than generalize it
- We initially tested with a complex model and noticed that even with many layers, the model didn’t perform well on noisy or unbalanced data.
- In AI model training, the most important thing is data preparation.That is why we used fewer layers with the cleanest data possible.

Improvements

Implement Cross-Validation

Instead of using a fixed validation set, rotate the validation set using k-fold cross-validation for more reliable performance measurement.

Expand the Dataset

Collect more images per class with diverse lighting, angles, and backgrounds.
Use controlled environments to reduce noise while still covering variety.

Improve Data Labeling & Cleaning

Use tools to detect mislabeled or blurry images.
Apply image enhancement techniques like sharpening or contrast correction.

Thankyou

@reallygreatsite