```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re
import pandas as pd

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!unzip "/content/drive/MyDrive/MS_in_AAI/Courses_and_Projects/AAI-590-IN1_Capstone Project/Capstone_Project/Capstone_Codebase/Capstone Project Results/eda_and_preprocessing_results.zip"
```

```
Archive:  /content/drive/MyDrive/MS_in_AAI/Courses_and_Projects/AAI-590-IN1_Capstone Project/Capstone_Project/Capstone_Codebase/Capstone Project Results/eda_and_preprocessing_results.zip
   creating: kaggle/working/
   creating: kaggle/working/.virtual_documents/
  inflating: kaggle/working/eda_cluster_representatives.png
  inflating: kaggle/working/metadata.csv
  inflating: kaggle/working/coral_features.npy
  inflating: kaggle/working/valid_paths.csv
  inflating: kaggle/working/eda_metrics_corr.png
  inflating: kaggle/working/eda_brightness_saturation_by_cluster.png
  inflating: kaggle/working/eda_cluster_counts.png
  inflating: kaggle/working/eda_texture_by_cluster.png
  inflating: kaggle/working/eda_folder_counts_with_clipped.png
  inflating: kaggle/working/tsne_results.npy
  inflating: kaggle/working/cluster_color_metrics.csv
  inflating: kaggle/working/eda_folder_counts.png
  inflating: kaggle/working/coral_clusters.csv
  inflating: kaggle/working/eda_tsne_clusters.png
```

```
!unzip "/content/drive/MyDrive/MS_in_AAI/Courses_and_Projects/AAI-590-IN1_Capstone Project/Capstone_Project/Capstone_Codebase/Capstone Project Results/modeling_results.zip" -d "/content/eda_results"
```

```
Archive:  /content/drive/MyDrive/MS_in_AAI/Courses_and_Projects/AAI-590-IN1_Capstone Project/Capstone_Project/Capstone_Codebase/Capstone Project Results/modeling_results.zip
   creating: /content/eda_results/kaggle/working/
  inflating: /content/eda_results/kaggle/working/cluster_distribution_by_source.png

   creating: /content/eda_results/kaggle/working/.virtual_documents/
  inflating: /content/eda_results/kaggle/working/efficientnetv2s_coral_clusters.keras
  inflating:
```

```
/content/eda_results/kaggle/working/metadata_with_clusters.csv
  inflating:
/content/eda_results/kaggle/working/feature_logreg_pipeline.joblib
  inflating:
/content/eda_results/kaggle/working/simple_nn_classifier.h5
  inflating: /content/eda_results/kaggle/working/confusion_matrix.png

  inflating:
/content/eda_results/kaggle/working/efficientnetv2s_training_report_su
mmary.csv
  inflating:
/content/eda_results/kaggle/working/coral_dataset_with_clusters_and_me
trics.csv
   creating: /content/eda_results/kaggle/working/img_model_ckpt/
  inflating:
/content/eda_results/kaggle/working/img_model_ckpt/best_img_model.h5
```

```
!unzip "/content/drive/MyDrive/MS_in_AAI/Courses_and_Projects/AAI-590-
IN1_Capstone Project/Capstone_Project/Capstone_Codebase/Capstone
Project Results/modeling_results.zip" -d "/content/modeling_results"
```

```
Archive:  /content/drive/MyDrive/MS_in_AAI/Courses_and_Projects/AAI-
590-IN1_Capstone Project/Capstone_Project/Capstone_Codebase/Capstone
Project Results/modeling_results.zip
   creating: /content/modeling_results/kaggle/working/
  inflating:
/content/modeling_results/kaggle/working/cluster_distribution_by_sourc
e.png
   creating:
/content/modeling_results/kaggle/working/.virtual_documents/
  inflating:
/content/modeling_results/kaggle/working/efficientnetv2s_coral_cluster
s.keras
  inflating:
/content/modeling_results/kaggle/working/metadata_with_clusters.csv
  inflating:
/content/modeling_results/kaggle/working/feature_logreg_pipeline.jobli
b
  inflating:
/content/modeling_results/kaggle/working/simple_nn_classifier.h5
  inflating:
/content/modeling_results/kaggle/working/confusion_matrix.png
  inflating:
/content/modeling_results/kaggle/working/efficientnetv2s_training_repo
rt_summary.csv
  inflating:
/content/modeling_results/kaggle/working/coral_dataset_with_clusters_a
nd_metrics.csv
   creating: /content/modeling_results/kaggle/working/img_model_ckpt/
  inflating:
```

```
/content/modeling_results/kaggle/working/img_model_ckpt/best_img_model
.h5

!ls -lrt
"/content/modeling_results/kaggle/working/efficientnetv2s_training_rep
ort_summary.csv"

-rw-r--r-- 1 root root 1429 Nov 23 06:22
/content/modeling_results/kaggle/working/efficientnetv2s_training_repo
rt_summary.csv
```

1. How effective was your machine learning model(s) at learning the task? e.g. Did the model(s) overfit/underfit the training data? What does the training accuracy/loss curve(s) look like for your model(s)?

Load Training History

```
history_path =
"/content/eda_results/kaggle/working/efficientnetv2s_training_report_s
ummary.csv"

history = pd.read_csv(history_path)
history.head()

{"summary":"{\n  \"name\": \"history\",\n  \"rows\": 12,\n
\"fields\": [\n    {\n      \"column\": \"Unnamed: 0\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 12,\n        \"samples\": [\n
\"confusion_matrix\",\n          \"image_model_weighted_recall\",\n
\"n_samples\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"0\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 11,\n        \"samples\": [\n
\"0.8997202623294729\",\n          \"39044\",\n          \"[[1546, 0,
142, 0, 0], [0, 433, 0, 14, 54], [73, 0, 2490, 31, 66], [0, 10, 87,
761, 74], [0, 23, 94, 36, 1094]]\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"history"}

efficientnetv2s_training_report_summary =
pd.read_csv("/content/eda_results/kaggle/working/efficientnetv2s_train
```

```
ing_report_summary.csv")
efficientnetv2s_training_report_summary
```


```
{"summary":"{\n  \"name\":
\"efficientnetv2s_training_report_summary\",\n  \"rows\": 12,\n
\"fields\": [\n    {\n      \"column\": \"Unnamed: 0\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 12,\n        \"samples\": [\n
\"confusion_matrix\",\n          \"image_model_weighted_recall\",\n
\"n_samples\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"0\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 11,\n        \"samples\": [\n
\"0.8997202623294729\",\n          \"39044\",\n          \"[[1546, 0,
142, 0, 0], [0, 433, 0, 14, 54], [73, 0, 2490, 31, 66], [0, 10, 87,
761, 74], [0, 23, 94, 36, 1094]]\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\
n}","type":"dataframe","variable_name":"efficientnetv2s_training_repor
t_summary"}
```


efficientnetv2s Training Plots

```
log_file = "/content/efficientnetv2s_training_log.txt"

with open(log_file, "r") as f:
    log = f.read()

# Regex to capture epoch-level summaries at the end of each epoch
pattern = re.compile(
    r"accuracy:\s*([0-9.]+)\s*-\s*loss:\s*([0-9.]+).*?"
    r"val_accuracy:\s*([0-9.]+)\s*-\s*val_loss:\s*([0-9.]+)",
    re.DOTALL
)

matches = pattern.findall(log)

epochs = []
train_acc = []
train_loss = []
val_acc = []
val_loss = []

for i, m in enumerate(matches):
    epochs.append(i+1)
    train_acc.append(float(m[0]))
    train_loss.append(float(m[1]))
    val_acc.append(float(m[2]))
    val_loss.append(float(m[3]))
```

```
df = pd.DataFrame({
    "epoch": epochs,
    "train_acc": train_acc,
    "train_loss": train_loss,
    "val_acc": val_acc,
    "val_loss": val_loss
})

df
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 6,\n  \"fields\": [\n{\n      \"column\": \"epoch\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 1,\n\"max\": 6,\n        \"num_unique_values\": 6,\n        \"samples\": [\n          1,\n          2,\n          6\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"train_acc\",\n\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.12370163162491703,\n        \"min\": 0.4604,\n        \"max\": 0.7959,\n        \"num_unique_values\": 6,\n        \"samples\": [\n0.4604,\n          0.6818,\n          0.7959\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"train_loss\",\n\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.497881046586431,\n        \"min\": 0.4917,\n        \"max\": 1.80537,\n        \"num_unique_values\": 6,\n        \"samples\": [\n1.80537,\n          0.781,\n          0.4917\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"val_acc\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.23720356447574728,\n        \"min\": 0.1844,\n        \"max\": 0.7002,\n        \"num_unique_values\": 6,\n        \"samples\": [\n0.1917,\n          0.6659,\n          0.7002\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"val_loss\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 1.1916320177247113,\n        \"min\": 0.8579,\n        \"max\": 4.0015,\n        \"num_unique_values\": 6,\n        \"samples\": [\n4.0015,\n          0.8579,\n          1.033\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


```
plt.figure(figsize=(12,5))
plt.plot(df["epoch"], df["train_acc"], label="Train Accuracy")
plt.plot(df["epoch"], df["val_acc"], label="Val Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Training vs Validation Accuracy")
plt.legend()
plt.grid(True)
```

```
plt.show()

plt.figure(figsize=(12,5))
plt.plot(df["epoch"], df["train_loss"], label="Train Loss")
plt.plot(df["epoch"], df["val_loss"], label="Val Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Training vs Validation Loss")
plt.legend()
plt.grid(True)
plt.show()
```

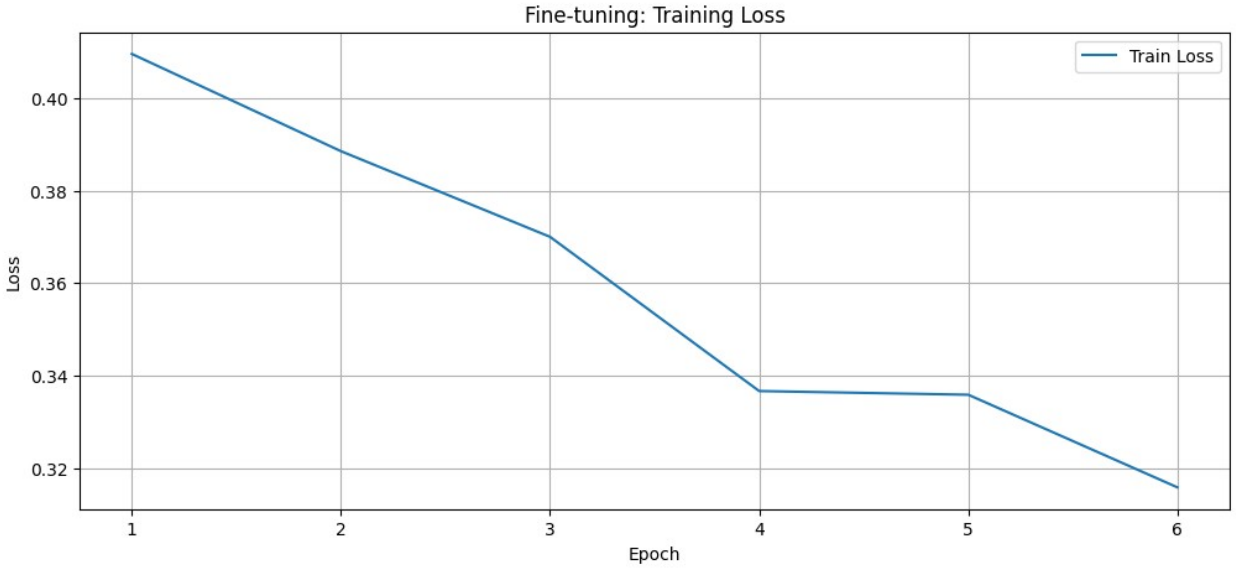
efficientnetv2s Finetuning Plots

```
log_file = "/content/efficientnetv2s_finetuning_log.txt"

with open(log_file, "r") as f:
    log = f.read()

# Regex to capture loss, acc, val_acc
pattern = re.compile(
    r"loss=([0-9.]+), acc=([0-9.]+), val_acc=([0-9.]+)"
)

matches = pattern.findall(log)

epochs = []
train_acc = []
train_loss = []
val_acc = []

for i, m in enumerate(matches):
    epochs.append(i+1)
    train_loss.append(float(m[0]))
    train_acc.append(float(m[1]))
    val_acc.append(float(m[2]))

df_ft = pd.DataFrame({
    "epoch": epochs,
    "train_acc": train_acc,
    "train_loss": train_loss,
    "val_acc": val_acc
})

df_ft
```


```
{"summary":"{\n  \"name\": \"df_ft\",\n  \"rows\": 6,\n  \"fields\":
[\n    {\n      \"column\": \"epoch\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 6,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          1,\n          2,\n          6\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"train_acc\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.01442674599485275,\n        \"min\": 0.8303,\n        \"max\":
0.8683,\n        \"num_unique_values\": 6,\n        \"samples\": [\n
0.8303,\n          0.838,\n          0.8683\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"train_loss\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.03585796517744232,\n        \"min\": 0.3159,\n        \"max\":
0.4095,\n        \"num_unique_values\": 6,\n        \"samples\": [\n
```

```
0.4095,\n          0.3885,\n          0.3159\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"val_acc\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.012044044171290642,\n        \"min\": 0.8597,\n        \"max\":
0.8957,\n        \"num_unique_values\": 6,\n        \"samples\": [\n
0.8786,\n          0.8832,\n          0.8715\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df_ft"}
```

```
plt.figure(figsize=(12,5))
plt.plot(df_ft["epoch"], df_ft["train_acc"], label="Train Accuracy")
plt.plot(df_ft["epoch"], df_ft["val_acc"], label="Val Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Fine-tuning: Training vs Validation Accuracy")
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(12,5))
plt.plot(df_ft["epoch"], df_ft["train_loss"], label="Train Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Fine-tuning: Training Loss")
plt.legend()
plt.grid(True)
plt.show()
```


Fine-tuning: Training Loss
Train Loss
Loss
0.40
0.38
0.36
0.34
0.32
1
2
3
4
5
6
Epoch