dl_model_analysis.ipynb

[ ]:
```
import json

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

plt.rcParams["figure.figsize"] = [8, 6]
```

[ ]:
```
with open("logs/evaluation.json", "r") as f:
    result = json.load(f)

result_train = []
backbones = {
    "resnet26d": "ResNet-26D",
    "mobilenetv3_small_100": "Mobile Net",
    "efficientnet_b0": "Efficient Net",
}
for backbone in backbones:
    result_train.append(pd.read_csv(f"logs/{backbone}.csv"))
    result_train[-1]["Model"] = backbones[backbone]
    epochs = []
    for i in range(200):
        epochs.extend([i + 1, i + 1])
    result_train[-1]["Epoch"] = epochs

result_train = pd.concat(result_train)
result_train[["Accuracy", "F1-Macro"]] *= 100
result_train[["Loss", "Accuracy", "F1-Macro"]] = result_train[
    ["Loss", "Accuracy", "F1-Macro"]
].round(2)
result_train
```

[ ]:
```
plt.figure()
sns.lineplot(data=result_train, x="Epoch", y="Loss", hue="Model", style="Subset")
plt.legend(loc="center right")
plt.tight_layout()
plt.savefig("logs/loss_dl_train.pdf")
plt.show()

plt.figure()
sns.lineplot(data=result_train, x="Epoch", y="Accuracy", hue="Model",␣
 ↪style="Subset")
plt.legend(loc="center right")
plt.tight_layout()
plt.savefig("logs/accuracy_dl_train.pdf")
plt.show()

plt.figure()
sns.lineplot(data=result_train, x="Epoch", y="F1-Macro", hue="Model",␣
 ↪style="Subset")
plt.legend(loc="center right")
plt.tight_layout()
plt.savefig("logs/f1_dl_train.pdf")
plt.show()
```

[ ]:
```
result_clean = []
for backbone in backbones:
    result_clean.append({"Model": backbones[backbone],␣
 ↪**result[backbone]["default"]})
result_clean = pd.DataFrame(result_clean)
result_clean[["Accuracy", "F1-Macro"]] *= 100
result_clean[["Loss", "Accuracy", "F1-Macro"]] = result_clean[
    ["Loss", "Accuracy", "F1-Macro"]
].round(2)
```

[ ]:
```
plt.figure()
ax = sns.barplot(data=result_clean, x="Model", y="Loss")
for p in ax.patches:
    ax.annotate(
        format(p.get_height(), ".2f"),
        (p.get_x() + p.get_width() / 2.0, p.get_height()),
        ha="center",
        va="center",
        xytext=(0, 9),
        textcoords="offset points",
    )
plt.savefig("logs/accuracy_loss.pdf")
plt.show()

plt.figure()
ax = sns.barplot(data=result_clean, x="Model", y="Accuracy")
for p in ax.patches:
    ax.annotate(
        format(p.get_height(), ".2f"),
        (p.get_x() + p.get_width() / 2.0, p.get_height()),
        ha="center",
        va="center",
        xytext=(0, 9),
        textcoords="offset points",
    )
plt.savefig("logs/accuracy_dl_clean.pdf")
```

```python
plt.figure()
ax = sns.barplot(data=result_clean, x="Model", y="F1-Macro")
for p in ax.patches:
    ax.annotate(
        format(p.get_height(), ".2f"),
        (p.get_x() + p.get_width() / 2.0, p.get_height()),
        ha="center",
        va="center",
        xytext=(0, 9),
        textcoords="offset points",
    )
plt.savefig("logs/f1_dl_clean.pdf")
```

[ ]:
```python
backbones_idx = {v: k for k, v in backbones.items()}
best_model = result_clean["F1-Macro"].idxmax()

result_perturbed = result[backbones_idx[result_clean["Model"][best_model]]]

data = []
for perturbation in result_perturbed:
    if perturbation == "default":
        continue
    name = " ".join(perturbation.split("_")).title()
    for value in result_perturbed[perturbation]:
        data.append(
            {
                "Perturbation": name,
                "Value": round(float(value), 2),
                "Loss": result_perturbed[perturbation][value]["Loss"],
                "Accuracy": result_perturbed[perturbation][value]["Accuracy"],
                "F1-Macro": result_perturbed[perturbation][value]["F1-Macro"],
            }
        )
result_perturbed = pd.DataFrame(data)
result_perturbed = result_perturbed.sort_values(["Perturbation", "Value"])
result_perturbed.to_csv("logs/resnet_perturbation.csv")
display(result_perturbed)
```

[ ]:
```python

```