# Hybrid Model Performance Analysis

August 7, 2025

## 0.1 Hybrid Model Evaluation

Jakob Balkovec Tue, Aug 5th 2025

This document evaluates the performance of the hybrid model, solely on metrics/numbers I got whilst training it. The model was trained on a dataset of 1841 images, and 184100 patches. Due to the stack mismatch issue, I didn't get a chance to filter out the noise (black patches) but I think the results are still quite promising.

The model is a dual-branch DL architecture (global and local), it leverages two ResNet18 backbones: one processes the full fundus image to extract global context, the other encodes localized lesion patches.

The output features orm each branch are then projected into a shared latent sapce via fully connected layers and fused using a multi-head self-attention mechanism…allowing the model to align and weigh patch-level signals relative to the image context.

The fused representation is passed through a lightweight MLP classifier with a sigmoid activation to predict the presence of four lesion types.

## 0.2 Training

The model was trained with a learning rate of 0.0001, and I used the Adam optimizer (work to be done here). The training loss decreased steadily over time, and the validation loss also showed a similar trend.

Since I couldn't access Texas A&M's GPU cluster, I had to train the model on Google Colab. I got booted off after about 9.5 epochs, but `f1` was still improving, meaning there is performance left on the table.

I designed a new training function that allows for early stopping as well as continuing training from a checkpoint. This way, I can resume training on the cluster without losing progress. I haven't tested any of this yet, but I'm hoping it works…

Google Colab for Education is not available in my country. I emailed Google support, but they said they can't help me. I'll make the most of what I have…

## 0.3 Data

[30]:
```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

```
import pandas as pd

CSV_FILE = "/Users/jbalkovec/Desktop/DR/experiments/model_logs/hybrid_model_log.
 ↪csv"
```

[31]:
```
df = pd.read_csv(CSV_FILE)
df.head(n=len(df))
```

[31]:

| | timestamp | epoch | f1_micro | hamming | subset_acc | iou | dice | pearson_r | loss |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2025-08-07 06:17:55 | 1 | 0.737753 | 0.241857 | 0.325733 | 0.683768 | 0.765708 | 0.507020 | 0.509871 |
| 1 | 2025-08-07 06:25:24 | 1 | 0.701047 | 0.238192 | 0.315418 | 0.689558 | 0.773851 | 0.556022 | NaN |
| 2 | 2025-08-07 06:35:14 | 2 | 0.754473 | 0.230185 | 0.345820 | 0.690147 | 0.769050 | 0.557325 | 0.474096 |
| 3 | 2025-08-07 06:42:39 | 2 | 0.800591 | 0.211726 | 0.386536 | 0.703131 | 0.775575 | 0.607541 | NaN |
| 4 | 2025-08-07 06:52:31 | 3 | 0.775229 | 0.212948 | 0.387622 | 0.707338 | 0.780621 | 0.595436 | 0.444653 |
| 5 | 2025-08-07 07:00:05 | 3 | 0.813652 | 0.197476 | 0.409338 | 0.727832 | 0.797764 | 0.630317 | NaN |
| 6 | 2025-08-07 07:10:09 | 4 | 0.802093 | 0.200190 | 0.394680 | 0.721544 | 0.792883 | 0.629872 | 0.421925 |
| 7 | 2025-08-07 07:17:46 | 4 | 0.836930 | 0.177796 | 0.461455 | 0.738916 | 0.800284 | 0.671015 | NaN |
| 8 | 2025-08-07 07:27:44 | 5 | 0.818863 | 0.184853 | 0.440282 | 0.734980 | 0.800610 | 0.658281 | 0.399331 |
| 9 | 2025-08-07 07:35:22 | 5 | 0.818492 | 0.167210 | 0.482628 | 0.757555 | 0.817199 | 0.711187 | NaN |
| 10 | 2025-08-07 07:45:18 | 6 | 0.827449 | 0.182546 | 0.454940 | 0.741450 | 0.804788 | 0.670248 | 0.388381 |

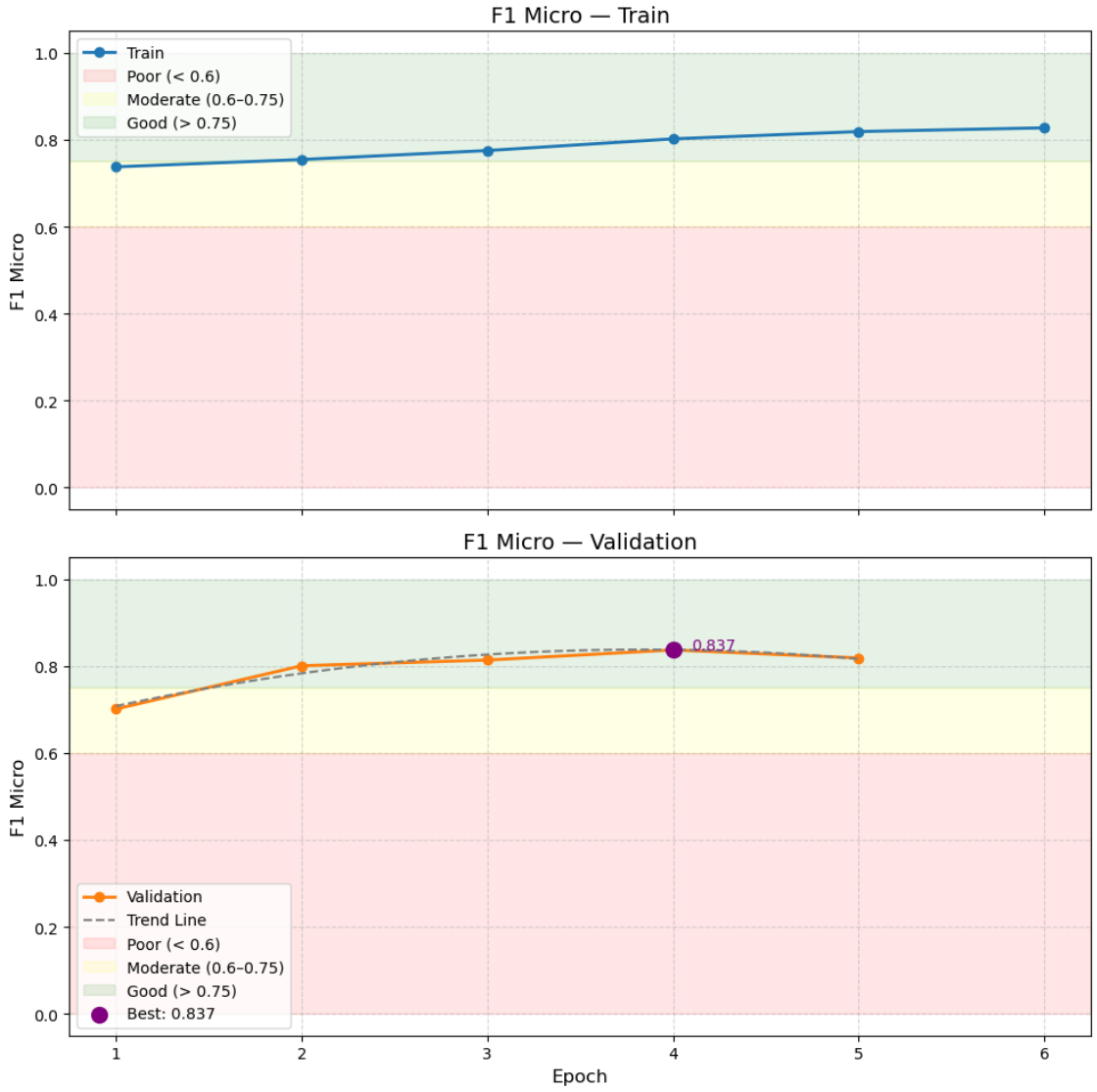
[32]:
```
def plot_metric(df, metric: str):

    # Split into train/val
    df_train = df.iloc[::2].copy()
    df_val = df.iloc[1::2].copy()

    fig, axes = plt.subplots(2, 1, figsize=(10, 10), sharex=True)

    # Plot TRAIN
    ax_train = axes[0]
    ax_train.plot(df_train["epoch"], df_train[metric], marker='o', color='tab:
 ↪blue', label='Train', linewidth=2)
    ax_train.set_ylabel(metric.replace('_', ' ').title(), fontsize=12)
```

```
    ax_train.set_title(f"{metric.replace('_', ' ').title()} - Train",␣
↪fontsize=14)
    ax_train.grid(True, linestyle='--', alpha=0.5)

    # Plot VAL
    ax_val = axes[1]
    ax_val.plot(df_val["epoch"], df_val[metric], marker='o', color='tab:
↪orange', label='Validation', linewidth=2)
    ax_val.set_xlabel("Epoch", fontsize=12)
    ax_val.set_ylabel(metric.replace('_', ' ').title(), fontsize=12)
    ax_val.set_title(f"{metric.replace('_', ' ').title()} - Validation",␣
↪fontsize=14)
    ax_val.grid(True, linestyle='--', alpha=0.5)

    # Trend line on VAL only
    if df_val[metric].notna().sum() >= 3:
        z = np.polyfit(df_val["epoch"], df_val[metric], 2)
        p = np.poly1d(z)
        x_vals = np.linspace(df_val["epoch"].min(), df_val["epoch"].max(), 100)
        ax_val.plot(x_vals, p(x_vals), "--", color="gray", label="Trend Line")

    # -------- THRESHOLDS --------
    def draw_thresholds(ax):
        if metric == 'f1_micro':
            ax.axhspan(0.0, 0.6, color='red', alpha=0.1, label="Poor (< 0.6)")
            ax.axhspan(0.6, 0.75, color='yellow', alpha=0.1, label="Moderate (0.
↪6-0.75)")
            ax.axhspan(0.75, 1.0, color='green', alpha=0.1, label="Good (> 0.
↪75)")
        elif metric == 'subset_acc':
            ax.axhspan(0.0, 0.4, color='red', alpha=0.1, label="Low (< 0.4)")
            ax.axhspan(0.4, 0.65, color='yellow', alpha=0.1, label="Moderate (0.
↪4-0.65)")
            ax.axhspan(0.65, 1.0, color='green', alpha=0.1, label="Good (> 0.
↪65)")
        elif metric == 'hamming':
            ax.axhspan(0.0, 0.15, color='green', alpha=0.1, label="Low Error (<␣
↪0.15)")
            ax.axhspan(0.15, 0.25, color='yellow', alpha=0.1, label="Moderate␣
↪Error (0.15-0.25)")
            ax.axhspan(0.25, 1.0, color='red', alpha=0.1, label="High Error (>␣
↪0.25)")
        elif metric == 'loss':
            ax.axhspan(0.0, 0.3, color='green', alpha=0.1, label="Low Loss (< 0.
↪3)")
```

```
            ax.axhspan(0.3, 0.6, color='yellow', alpha=0.1, label="Moderate␣
↪Loss (0.3-0.6)")
            ax.axhspan(0.6, 1.0, color='red', alpha=0.1, label="High Loss (> 0.
↪6)")
        elif metric == 'pearson_r':
            ax.axhspan(-1.0, 0.3, color='red', alpha=0.1, label="Weak (< 0.3)")
            ax.axhspan(0.3, 0.6, color='yellow', alpha=0.1, label="Moderate (0.
↪3-0.6)")
            ax.axhspan(0.6, 1.0, color='green', alpha=0.1, label="Strong (> 0.
↪6)")
        elif metric == 'dice':
            ax.axhspan(0.0, 0.6, color='red', alpha=0.1, label="Poor (< 0.6)")
            ax.axhspan(0.6, 0.75, color='yellow', alpha=0.1, label="Fair (0.6-0.
↪75)")
            ax.axhspan(0.75, 1.0, color='green', alpha=0.1, label="Good (> 0.
↪75)")
        elif metric == 'iou':
            ax.axhspan(0.0, 0.5, color='red', alpha=0.1, label="Low (< 0.5)")
            ax.axhspan(0.5, 0.7, color='yellow', alpha=0.1, label="Moderate (0.
↪5-0.7)")
            ax.axhspan(0.7, 1.0, color='green', alpha=0.1, label="High (> 0.7)")

    draw_thresholds(ax_train)
    draw_thresholds(ax_val)

    # -------- BEST ---------
    df_val_clean = df_val[df_val[metric].notna()]
    if not df_val_clean.empty:
        if metric in ['loss', 'hamming']:
            best_idx = df_val_clean[metric].idxmin()
        else:
            best_idx = df_val_clean[metric].idxmax()

        best_epoch = df_val_clean.loc[best_idx, "epoch"]
        best_value = df_val_clean.loc[best_idx, metric]
        ax_val.scatter(best_epoch, best_value, color='purple', s=100, zorder=5,␣
↪label=f"Best: {best_value:.3f}")
        ax_val.text(best_epoch + 0.1, best_value, f"{best_value:.3f}",␣
↪color='purple', fontsize=10)

    # Legends
    ax_train.legend()
    ax_val.legend()

    plt.tight_layout()
    plt.show()
```

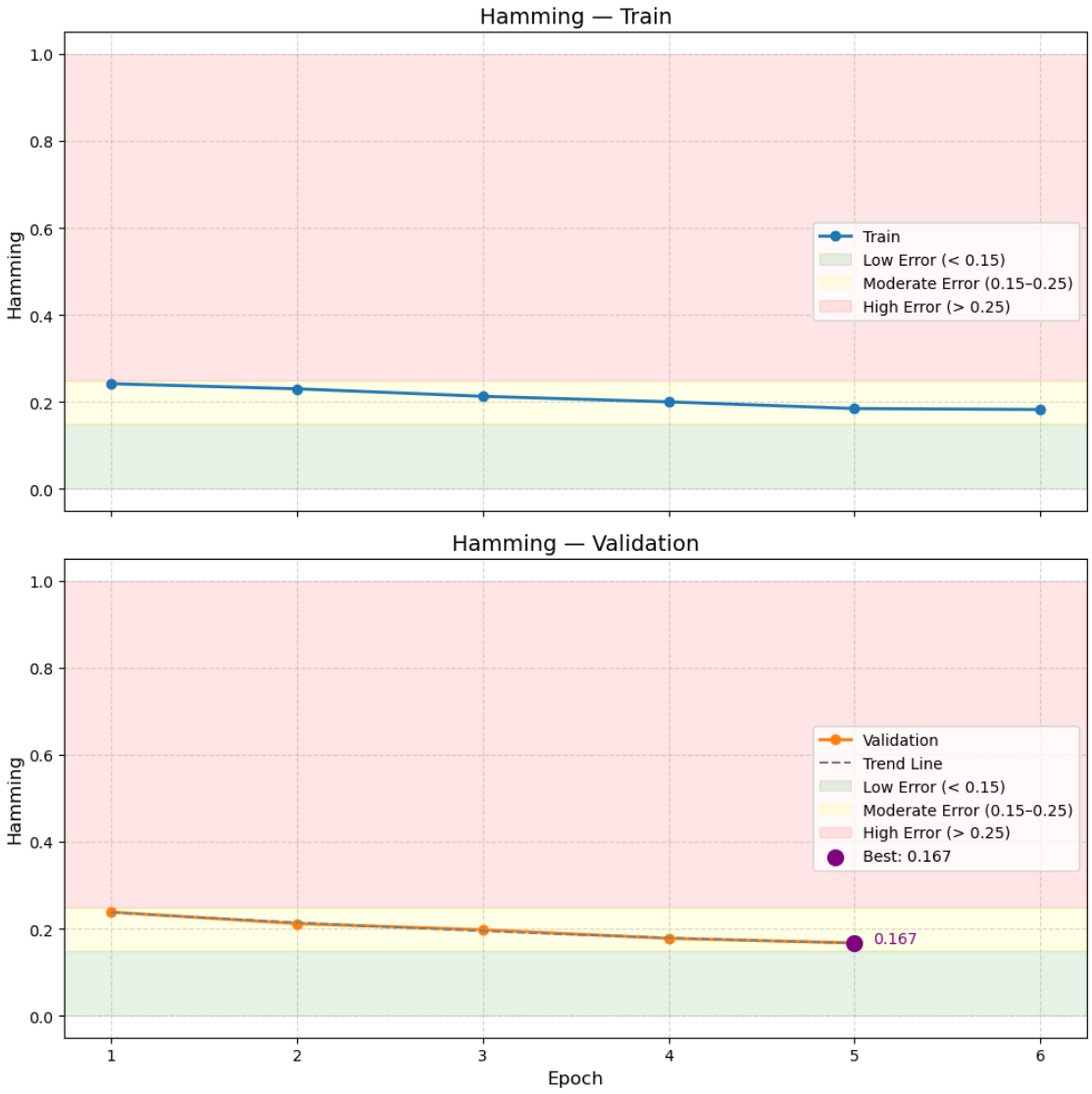
## 0.4 F1

[33]:
```
plot_metric(df, "f1_micro")
```



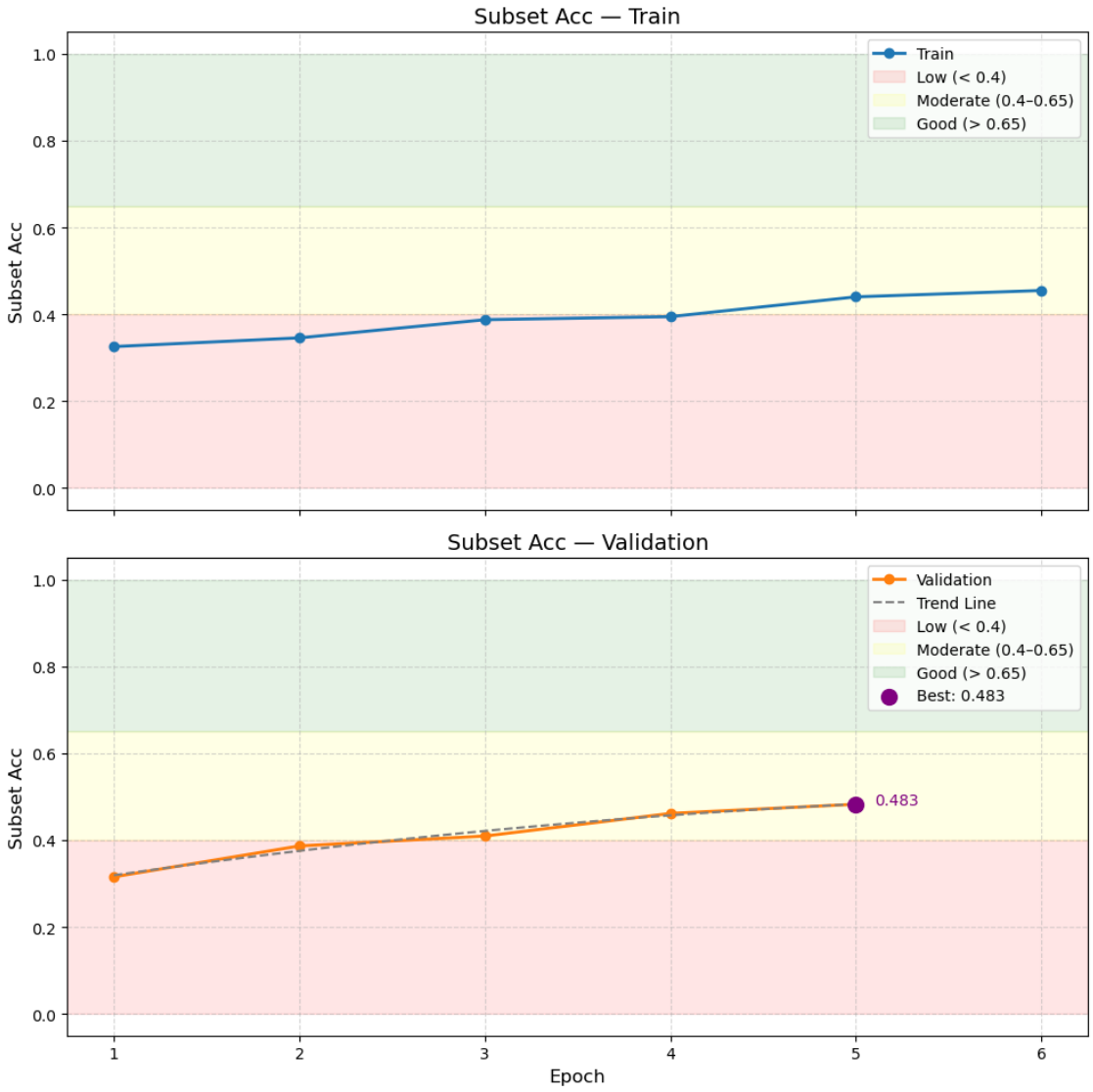

## 0.5 Hamming Loss

[34]:
```
plot_metric(df, "hamming")
```



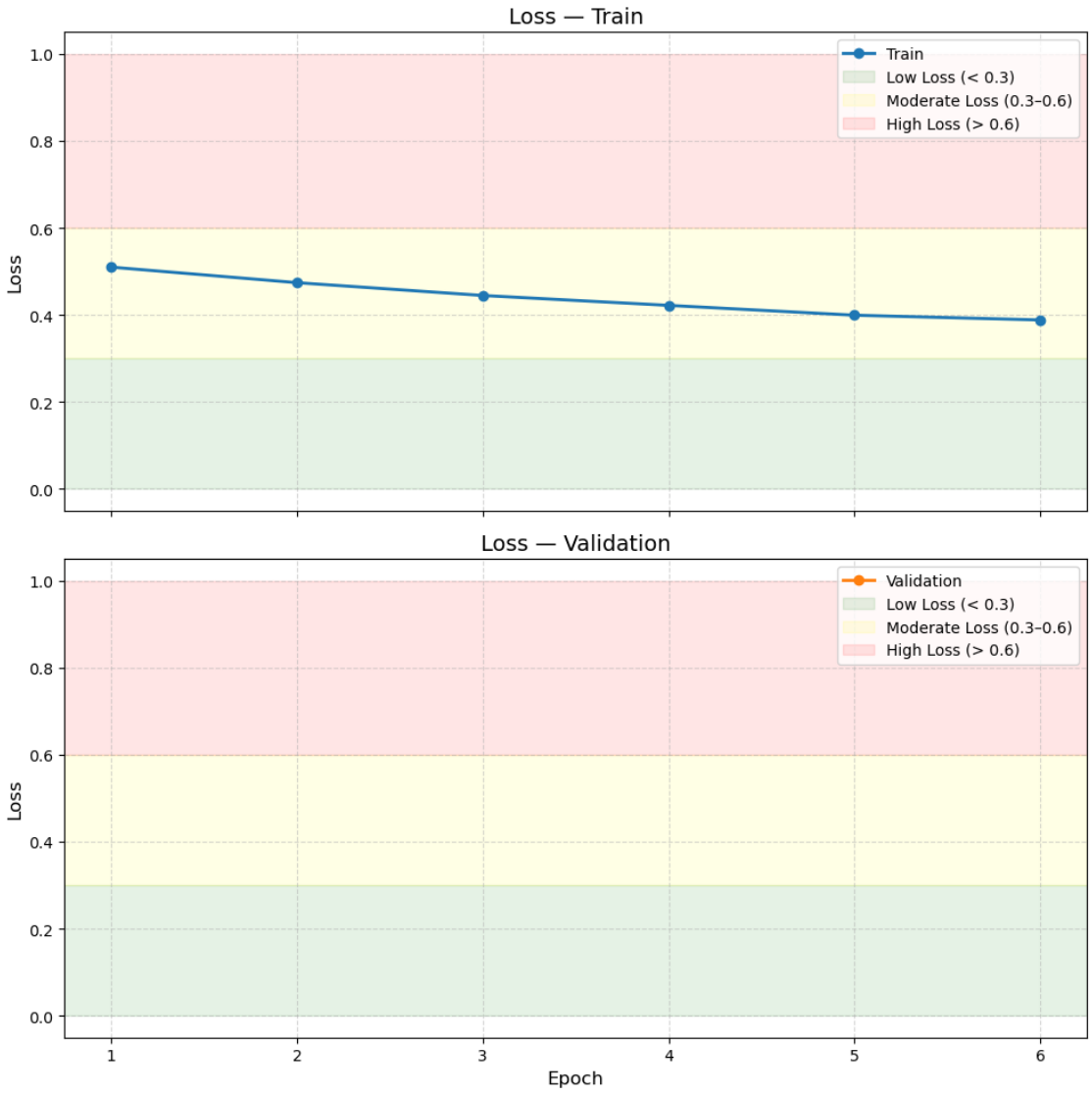

## 0.6 Subset Accuracy

[35]:
```
plot_metric(df, "subset_acc")
```



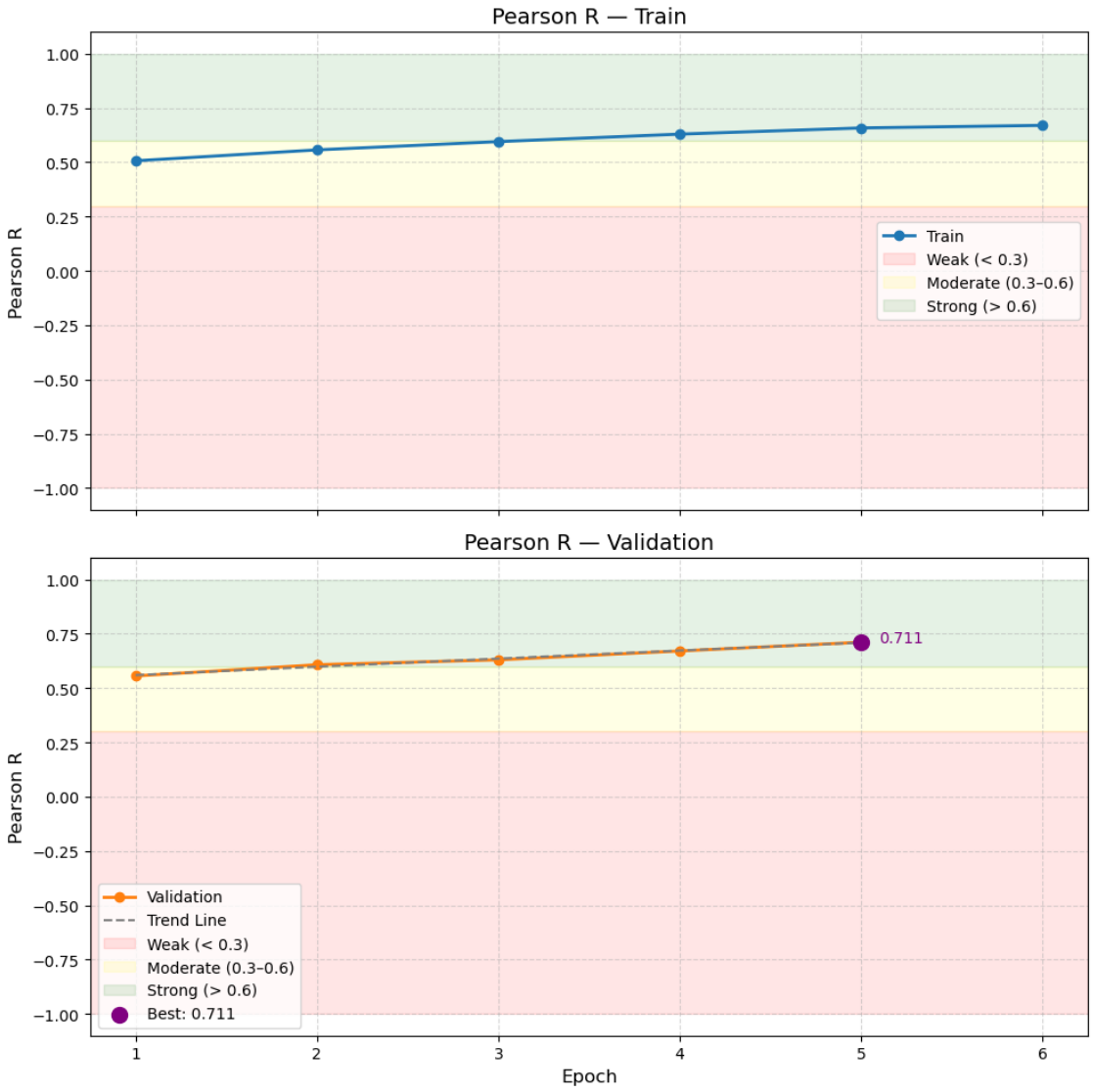

## 0.7 Loss

[36]:
```
plot_metric(df, "loss")
```



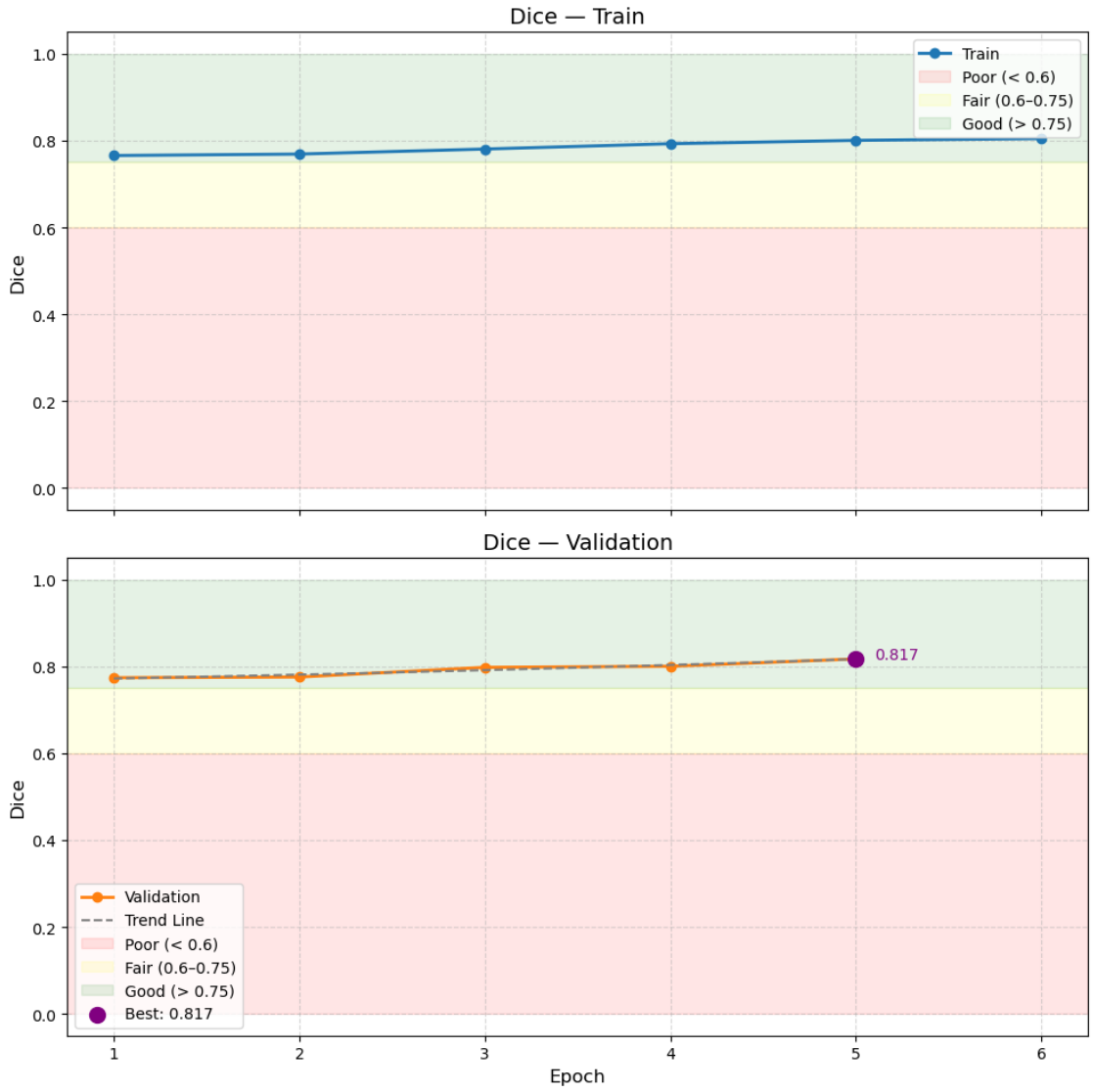

## 0.8 Pearson Correlation (R)

[37]:
```
plot_metric(df, "pearson_r")
```



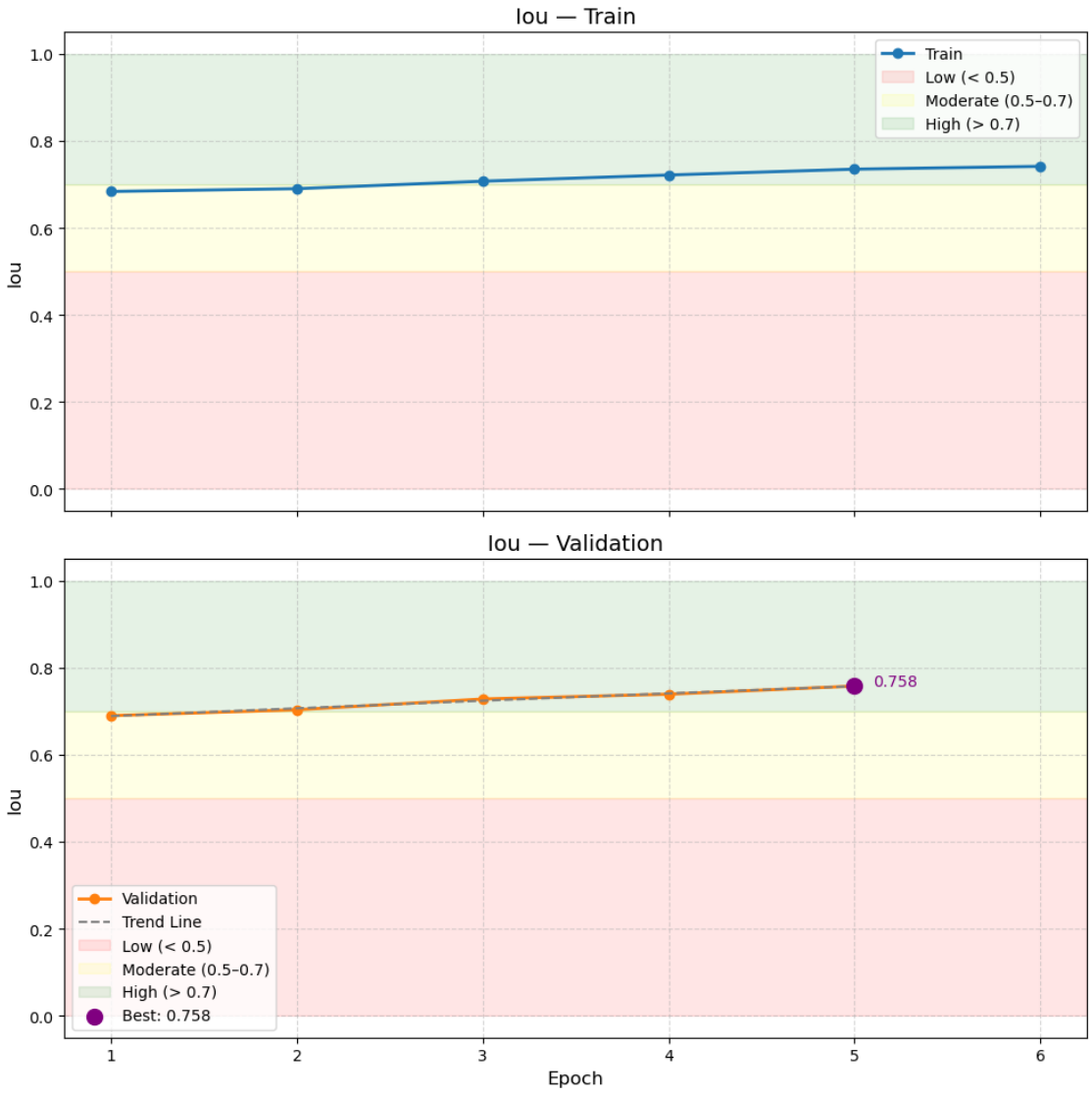

## 0.9 Dice

[38]:
```
plot_metric(df, "dice")
```




## 0.10 IoU

[39]:
```
plot_metric(df, "iou")
```


Iou — Train
Train
Low (< 0.5)
Moderate (0.5–0.7)
High (> 0.7)
Iou
1.0
0.8
0.6
0.4
0.2
0.0
Iou — Validation
0.758
Validation
Trend Line
Low (< 0.5)
Moderate (0.5–0.7)
High (> 0.7)
Best: 0.758
1
2
3
4
5
6
Epoch