analysisV3

August 17, 2025

0.1 Hybrid Model Evaluation

Jakob Balkovec Fri, Aug 15th 2025

0.2 Data

[1]:
```python
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd

METRICS_LOG_CSV = "metrics_log_latest.csv"
NPZ_DIR = "npz"
MASK_VISUALS_DIR = "mask_visuals"

LOSS_LAMBDA = 1.0
```

[2]:
```python
df = pd.read_csv(METRICS_LOG_CSV)
df.head()
```

[2]:
```
   epoch        lr  train_loss    val_loss  train_f1_micro  train_f1_macro  \
0      1  0.000010    0.679842   19.476119        0.898617        0.886766
1      2  0.000010    0.667445   38.415935        0.915791        0.903697
2      3  0.000010    0.517392   52.851267        0.983686        0.979669
3      4  0.000005    0.564369   87.611669        0.997005        0.995839
4      5  0.000005    0.637261  108.560271        0.998608        0.997989

   train_precision_micro  train_recall_micro  train_iou_micro  \
0               0.816809            0.998634         0.815898
1               0.845540            0.998772         0.844662
2               0.969635            0.998152         0.967897
3               0.995312            0.998703         0.994027
4               0.997714            0.999503         0.997220

   train_roc_auc_macro  …  train_coord_r2  val_coord_mae  val_coord_rmse  \
0                  NaN  …       -4.600443       0.325995        0.400923
1                  NaN  …       -4.840408       0.315038        0.386400
2                  NaN  …       -5.996484       0.283642        0.350308
```

```
3                  NaN  ...       -6.884182       0.300091        0.366335
4                  NaN  ...       -7.215684       0.301701        0.368077

   val_coord_r2  train_seg_iou  train_seg_dice  val_seg_iou  val_seg_dice  \
0     -1.310547       0.021654        0.042390     0.003729      0.007431
1     -1.146186       0.025959        0.050604     0.004036      0.008040
2     -0.763979       0.026841        0.052279     0.004305      0.008572
3     -0.929079       0.028381        0.055196     0.004210      0.008385
4     -0.947465       0.029913        0.058089     0.003814      0.007599

        monitor  monitor_value
0  val_f1_micro       0.756196
1  val_f1_micro       0.683835
2  val_f1_micro       0.681402
3  val_f1_micro       0.681213
4  val_f1_micro       0.681213

[5 rows x 60 columns]
```

[3]:
```python
def split_train_val(df):
    base = pd.DataFrame({"epoch": df["epoch"]})

    train_cols = [c for c in df.columns if c.startswith("train_")]
    val_cols   = [c for c in df.columns if c.startswith("val_")]

    def strip_prefix(cols, prefix):
        out = {}
        for c in cols:
            key = c[len(prefix):]
            out[key] = c
        return out

    train_map = strip_prefix(train_cols, "train_")
    val_map   = strip_prefix(val_cols, "val_")

    shared = sorted(set(train_map.keys()) & set(val_map.keys()))

    df_train = base.copy()
    df_val   = base.copy()
    for k in shared:
        df_train[k] = df[train_map[k]]
        df_val[k]   = df[val_map[k]]

    return df_train, df_val, shared

df_train, df_val, shared_metrics = split_train_val(df)
```

[4]:
```
df_train.head()
```

[4]:
```
   epoch  coord_mae  coord_r2  coord_rmse  f1_class_0  f1_class_1  f1_class_2  \
0      1   0.510571 -4.600443    0.622578    0.874932    0.994936    0.965306
1      2   0.519847 -4.840408    0.635766    0.885020    0.996473    0.971634
2      3   0.569415 -5.996484    0.695858    0.975304    0.999136    0.994531
3      4   0.600906 -6.884182    0.738680    0.997633    0.999933    0.999546
4      5   0.613254 -7.215684    0.754053    0.999339    0.999978    0.999916

   f1_class_3  f1_macro  f1_micro  …  precision_class_3  precision_micro  \
0    0.711892  0.886766  0.898617  …           0.553359         0.816809
1    0.761660  0.903697  0.915791  …           0.617613         0.845540
2    0.949704  0.979669  0.983686  …           0.912570         0.969635
3    0.986244  0.995839  0.997005  …           0.979902         0.995312
4    0.992722  0.997989  0.998608  …           0.988414         0.997714

   recall_class_0  recall_class_1  recall_class_2  recall_class_3  \
0        0.999166        0.997686        0.999713        0.997738
1        0.999626        1.000000        0.999857        0.993339
2        0.999195        1.000000        0.999976        0.989987
3        0.999684        1.000000        0.999952        0.992669
4        0.999942        1.000000        1.000000        0.997068

   recall_micro  roc_auc_macro  seg_dice   seg_iou
0      0.998634            NaN  0.042390  0.021654
1      0.998772            NaN  0.050604  0.025959
2      0.998152            NaN  0.052279  0.026841
3      0.998703            NaN  0.055196  0.028381
4      0.999503            NaN  0.058089  0.029913

[5 rows x 29 columns]
```

[5]:
```
df_val.head()
```

[5]:
```
   epoch  coord_mae  coord_r2  coord_rmse  f1_class_0  f1_class_1  f1_class_2  \
0      1   0.325995 -1.310547    0.400923    0.876564    0.867433    0.600865
1      2   0.315038 -1.146186    0.386400    0.876564    0.867433    0.027227
2      3   0.283642 -0.763979    0.350308    0.876564    0.867433    0.001957
3      4   0.300091 -0.929079    0.366335    0.876564    0.867433    0.000000
4      5   0.301701 -0.947465    0.368077    0.876564    0.867433    0.000000

   f1_class_3  f1_macro  f1_micro  …  precision_class_3  precision_micro  \
0         0.0  0.586215  0.756196  …                0.0         0.782517
1         0.0  0.442806  0.683835  …                0.0         0.774027
2         0.0  0.436488  0.681402  …                0.0         0.773150
3         0.0  0.435999  0.681213  …                0.0         0.773076
4         0.0  0.435999  0.681213  …                0.0         0.773076
```

```
   recall_class_0  recall_class_1  recall_class_2  recall_class_3  \
0             1.0             1.0        0.469916             0.0
1             1.0             1.0        0.013806             0.0
2             1.0             1.0        0.000979             0.0
3             1.0             1.0        0.000000             0.0
4             1.0             1.0        0.000000             0.0

   recall_micro  roc_auc_macro  seg_dice   seg_iou
0      0.731588            NaN  0.007431  0.003729
1      0.612469            NaN  0.008040  0.004036
2      0.609119            NaN  0.008572  0.004305
3      0.608863            NaN  0.008385  0.004210
4      0.608863            NaN  0.007599  0.003814

[5 rows x 29 columns]
```

[ ]:
```python
def LOSS_BANDS(loss_lambda: float = 1.0):
    low_thr  = 0.7 + 0.4 * loss_lambda
    high_thr = 1.5 + 0.4 * loss_lambda
    max_thr  = high_thr + 1.0
    return [
        (0.00,   low_thr,  "green",  f"Low (< {low_thr:.2f})"),
        (low_thr, high_thr,"yellow", f"Moderate ({low_thr:.2f}-{high_thr:.
↪2f})"),
        (high_thr, max_thr,"red",    f"High (> {high_thr:.2f})"),
    ]

BANDS = {
    "f1_micro": [
        (0.00, 0.60, "red",    "Poor (< 0.60)"),
        (0.60, 0.75, "yellow", "Moderate (0.60-0.75)"),
        (0.75, 1.00, "green",  "Good (> 0.75)"),
    ],
    "f1_macro": [
        (0.00, 0.60, "red",    "Poor (< 0.60)"),
        (0.60, 0.75, "yellow", "Moderate (0.60-0.75)"),
        (0.75, 1.00, "green",  "Good (> 0.75)"),
    ],
    "precision_micro": [
        (0.00, 0.60, "red",    "Low (< 0.60)"),
        (0.60, 0.80, "yellow", "Moderate (0.60-0.80)"),
        (0.80, 1.00, "green",  "High (> 0.80)"),
    ],
    "recall_micro": [
        (0.00, 0.60, "red",    "Low (< 0.60)"),
        (0.60, 0.80, "yellow", "Moderate (0.60-0.80)"),
```

```python
        (0.80, 1.00, "green",  "High (> 0.80)"),
    ],
    "iou_micro": [
        (0.00, 0.50, "red",    "Low (< 0.50)"),
        (0.50, 0.70, "yellow", "Moderate (0.50-0.70)"),
        (0.70, 1.00, "green",  "High (> 0.70)"),
    ],
    "roc_auc_macro": [
        (0.50, 0.70, "red",    "Below target (< 0.70)"),
        (0.70, 0.85, "yellow", "Decent (0.70-0.85)"),
        (0.85, 1.00, "green",  "Strong (> 0.85)"),
    ],

    "seg_iou": [
        (0.00, 0.50, "red",    "Low (< 0.50)"),
        (0.50, 0.70, "yellow", "Moderate (0.50-0.70)"),
        (0.70, 1.00, "green",  "High (> 0.70)"),
    ],
    "seg_dice": [
        (0.00, 0.60, "red",    "Poor (< 0.60)"),
        (0.60, 0.75, "yellow", "Fair (0.60-0.75)"),
        (0.75, 1.00, "green",  "Good (> 0.75)"),
    ],

    "coord_r2": [
        (-5.00, 0.00, "red",    "Worse than baseline (< 0)"),
        ( 0.00, 0.50, "yellow", "Moderate (0-0.5)"),
        ( 0.50, 1.00, "green",  "Good (> 0.5)"),
    ],
    "coord_mae": [
        (0.20, 10.0, "red",    "High error (> 0.20)"),
        (0.10, 0.20, "yellow", "Moderate (0.10-0.20)"),
        (0.00, 0.10, "green",  "Low (< 0.10)"),
    ],
    "coord_rmse": [
        (0.25, 10.0, "red",    "High error (> 0.25)"),
        (0.15, 0.25, "yellow", "Moderate (0.15-0.25)"),
        (-5.0, 0.15, "green",  "Low (< 0.15)"),
    ],
}

for c in range(4):
    BANDS[f"f1_class_{c}"] = [
        (0.00, 0.60, "red",    "Poor (< 0.60)"),
        (0.60, 0.75, "yellow", "Moderate (0.60-0.75)"),
        (0.75, 1.00, "green",  "Good (> 0.75)"),
    ]
```

```python
    BANDS[f"precision_class_{c}"] = [
        (0.00, 0.60, "red",    "Low (< 0.60)"),
        (0.60, 0.80, "yellow", "Moderate (0.60-0.80)"),
        (0.80, 1.00, "green",  "High (> 0.80)"),
    ]
    BANDS[f"recall_class_{c}"] = [
        (0.00, 0.60, "red",    "Low (< 0.60)"),
        (0.60, 0.80, "yellow", "Moderate (0.60-0.80)"),
        (0.80, 1.00, "green",  "High (> 0.80)"),
    ]
    BANDS[f"iou_class_{c}"] = [
        (0.00, 0.50, "red",    "Low (< 0.50)"),
        (0.50, 0.70, "yellow", "Moderate (0.50-0.70)"),
        (0.70, 1.00, "green",  "High (> 0.70)"),
    ]

def get_bands(metric_name: str, *, loss_lambda: float = 1.0):
    if metric_name == "loss":
        return LOSS_BANDS(loss_lambda)
    return BANDS.get(metric_name, None)
```

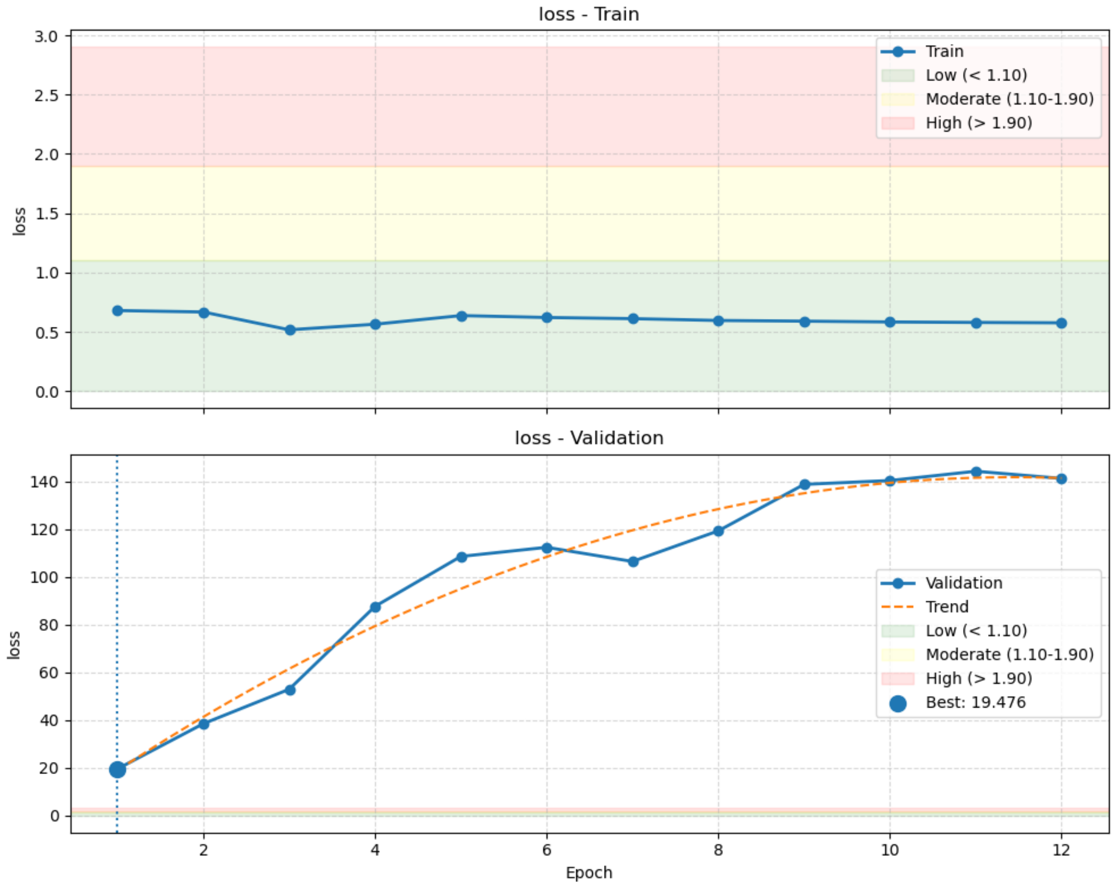
[ ]:
```python
def plot_train_val(df, metric_name, show_trend=True):
    train_col = f"train_{metric_name}"
    val_col = f"val_{metric_name}"
    if train_col not in df.columns or val_col not in df.columns:
        raise ValueError(f"{metric_name} not found in DataFrame.")

    x = df["epoch"]
    train_y = df[train_col]
    val_y = df[val_col]

    fig, axes = plt.subplots(2, 1, figsize=(10, 8), sharex=True)

    axes[0].plot(x, train_y, marker='o', label="Train", linewidth=2)
    axes[0].set_ylabel(metric_name)
    axes[0].set_title(f"{metric_name} - Train")
    axes[0].grid(True, linestyle="--", alpha=0.5)

    axes[1].plot(x, val_y, marker='o', label="Validation", linewidth=2)
    axes[1].set_xlabel("Epoch")
    axes[1].set_ylabel(metric_name)
    axes[1].set_title(f"{metric_name} - Validation")
    axes[1].grid(True, linestyle="--", alpha=0.5)

    if show_trend and val_y.notna().sum() >= 3:
        z = np.polyfit(x, val_y, 2)
        p = np.poly1d(z)
```

```
        x_fit = np.linspace(x.min(), x.max(), 200)
        axes[1].plot(x_fit, p(x_fit), "--", label="Trend")

    bands = get_bands(metric_name, loss_lambda=1.0)
    for ax in axes:
        if bands:
            for lo, hi, color, lab in bands:
                ax.axhspan(lo, hi, color=color, alpha=0.10, label=lab)

    val_clean = val_y.dropna()
    if not val_clean.empty:
        if metric_name in ["loss"]:
            best_idx = val_clean.idxmin()
        else:
            best_idx = val_clean.idxmax()
        best_epoch = x[best_idx]
        best_value = val_clean.loc[best_idx]
        axes[1].scatter(best_epoch, best_value, s=100, zorder=5, label=f"Best:␣
↪{best_value:.3f}")
        axes[1].axvline(best_epoch, linestyle=":", linewidth=1.5)

    for ax in axes:
        handles, labels = ax.get_legend_handles_labels()
        by_label = dict(zip(labels, handles))
        ax.legend(by_label.values(), by_label.keys(), loc="best")

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

1 Global Classification Metrics

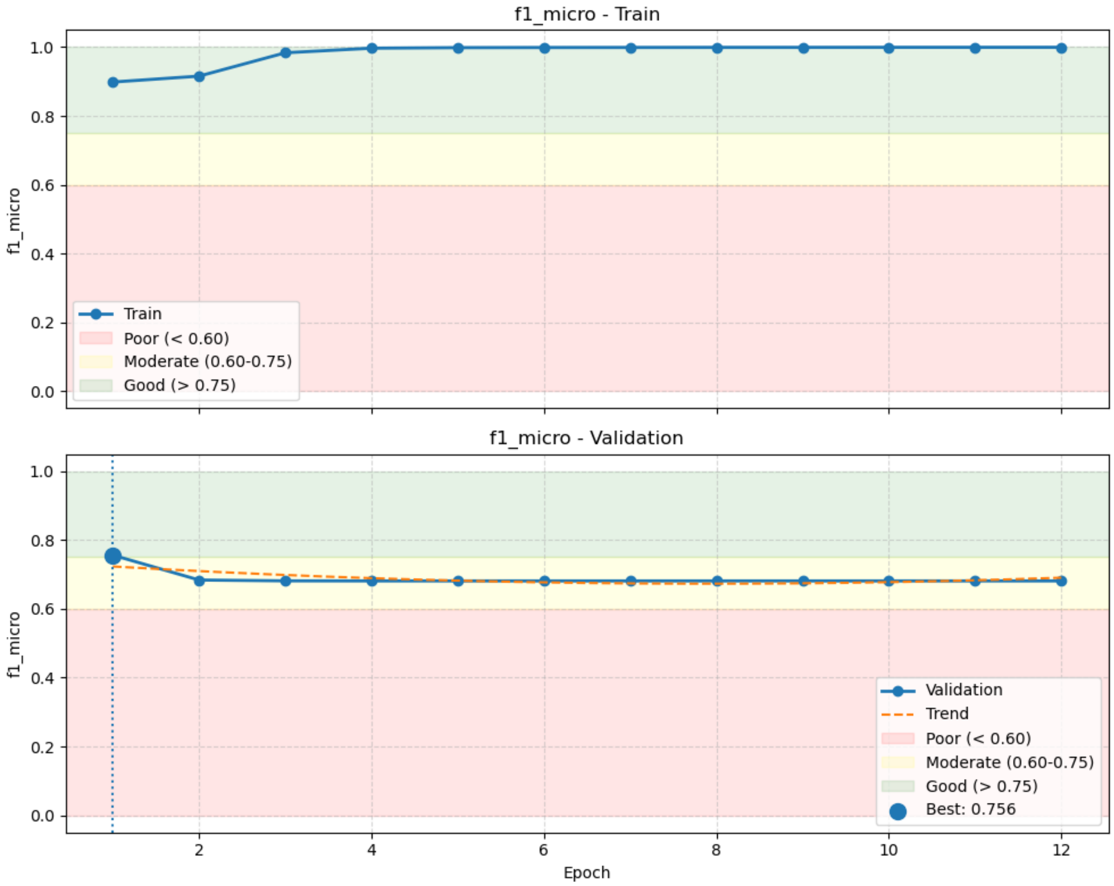
1.1 Loss

[8]:
```
plot_train_val(df, "loss") # fix scale
```



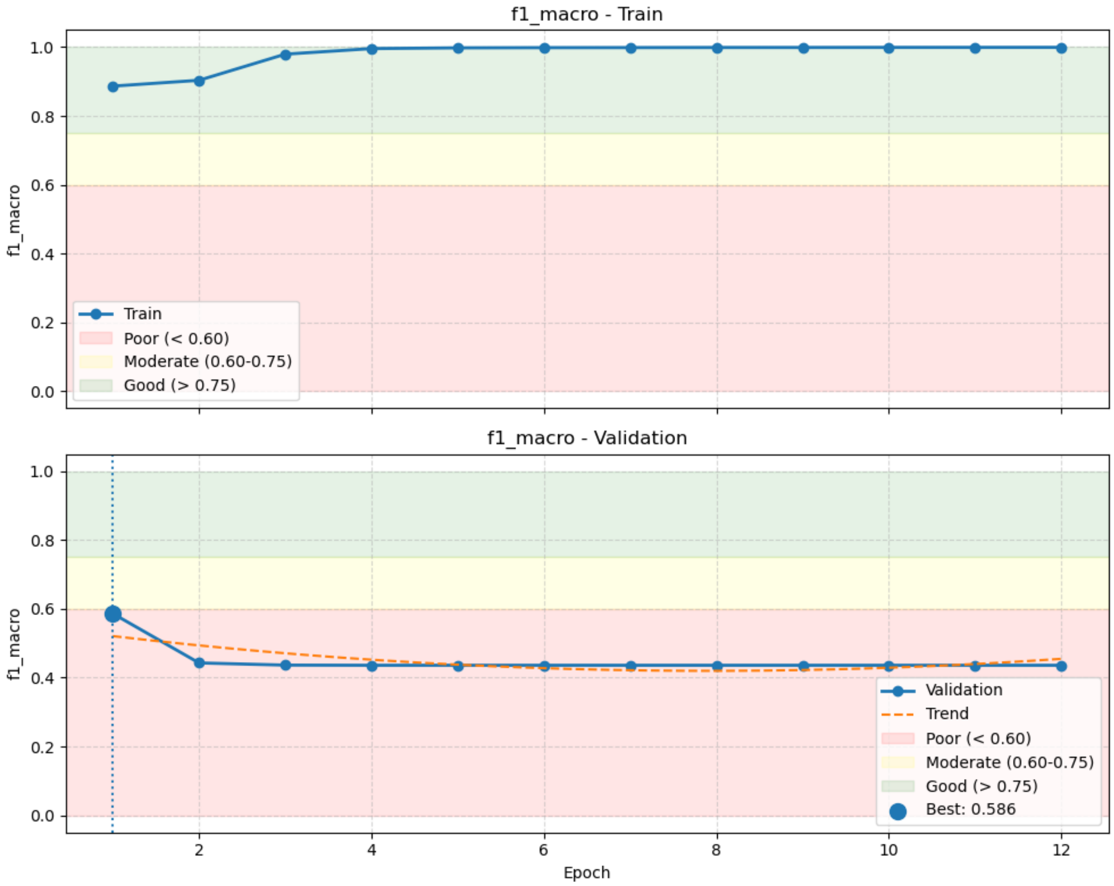
1.2 F1 - Micro

[9]:
```
plot_train_val(df, "f1_micro")
```



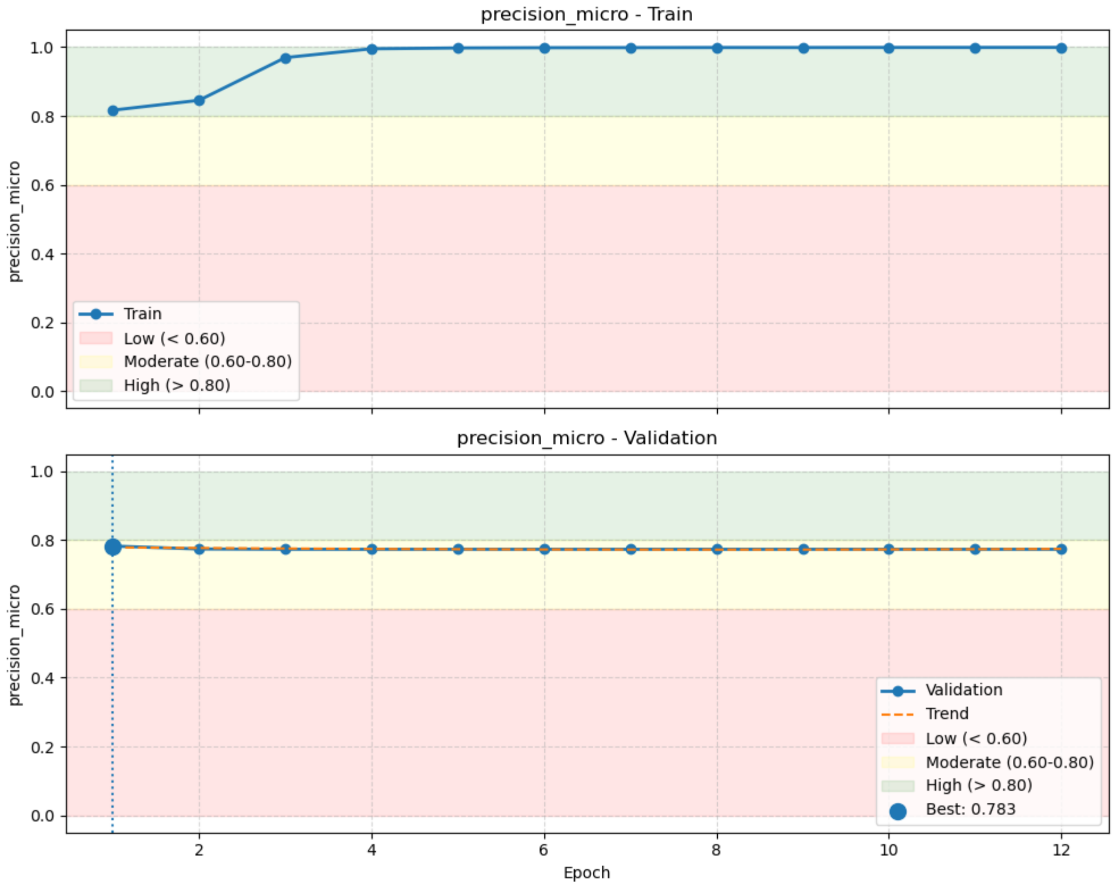
1.3 F1 - Macro

[10]:
```
plot_train_val(df, "f1_macro")
```



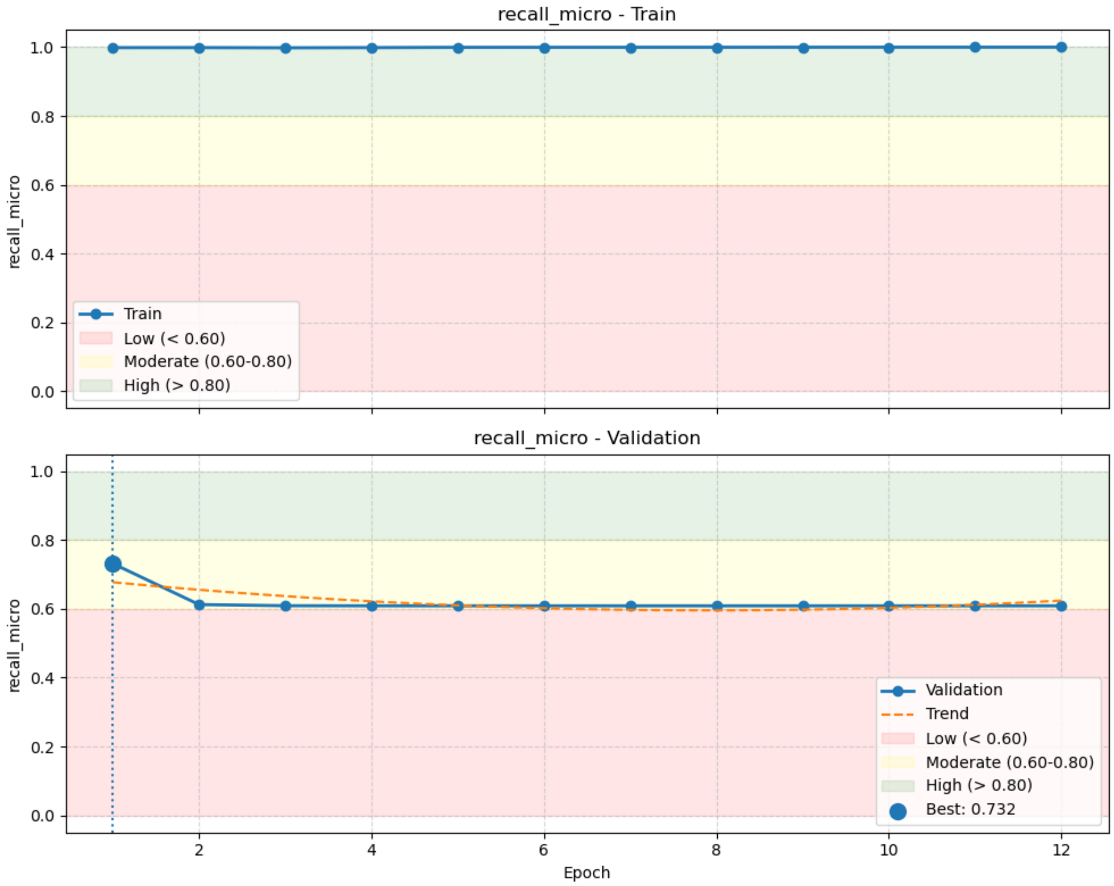
1.4 Percision Micro

[11]:
```
plot_train_val(df, "precision_micro")
```



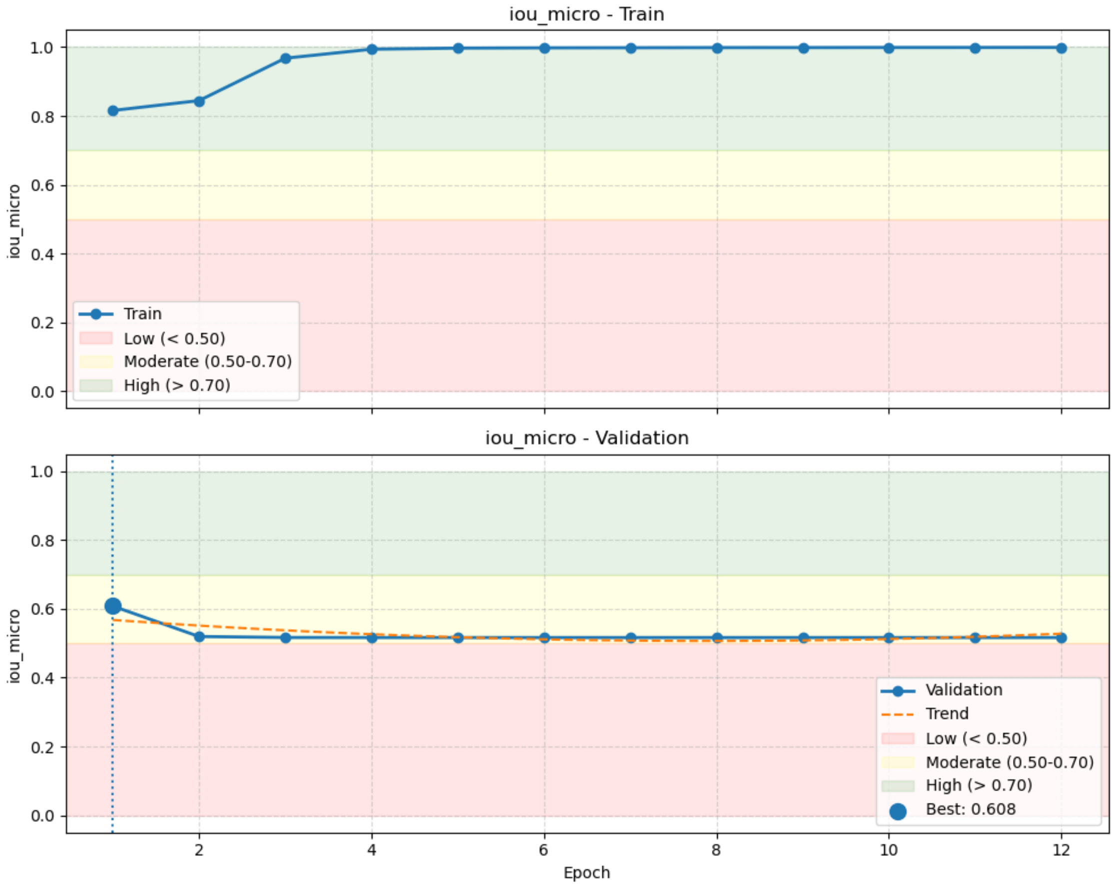
1.5 Recall Micro

[12]:
```
plot_train_val(df, "recall_micro")
```



1.6 IoU Micro

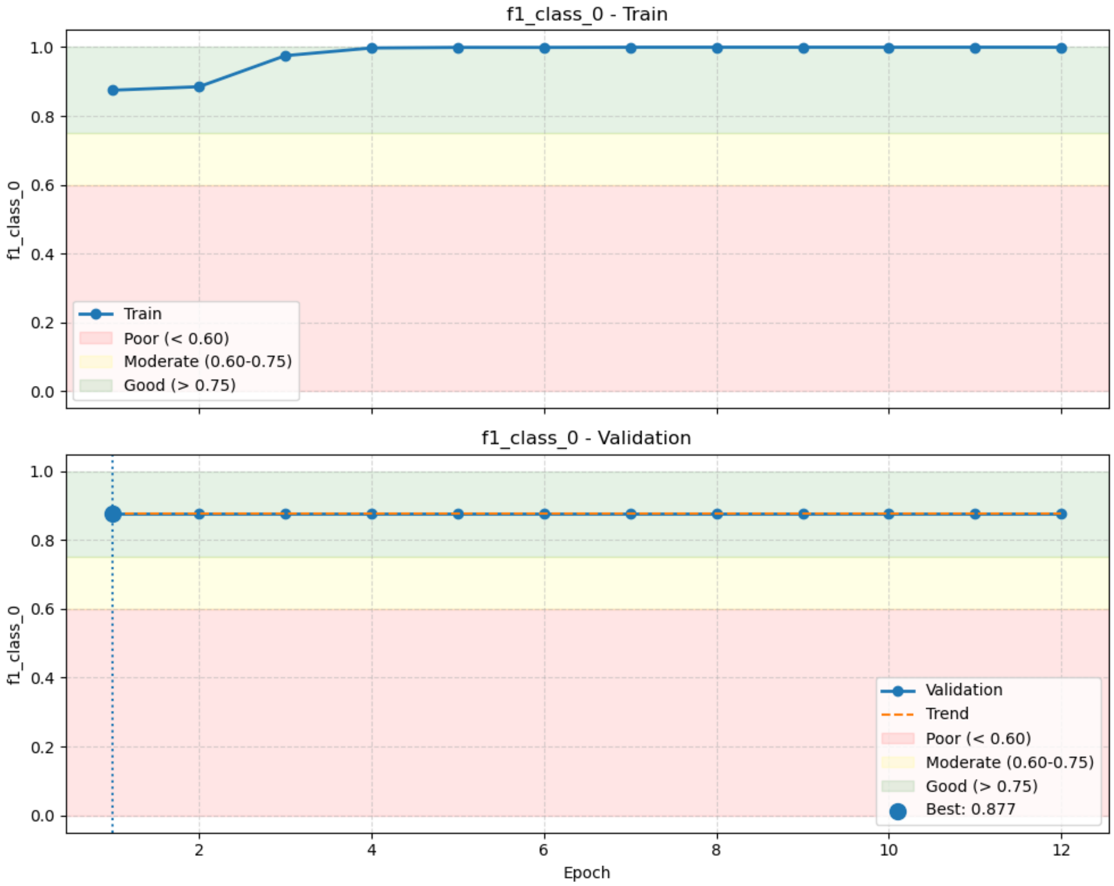
[13]:
```
plot_train_val(df, "iou_micro")
```



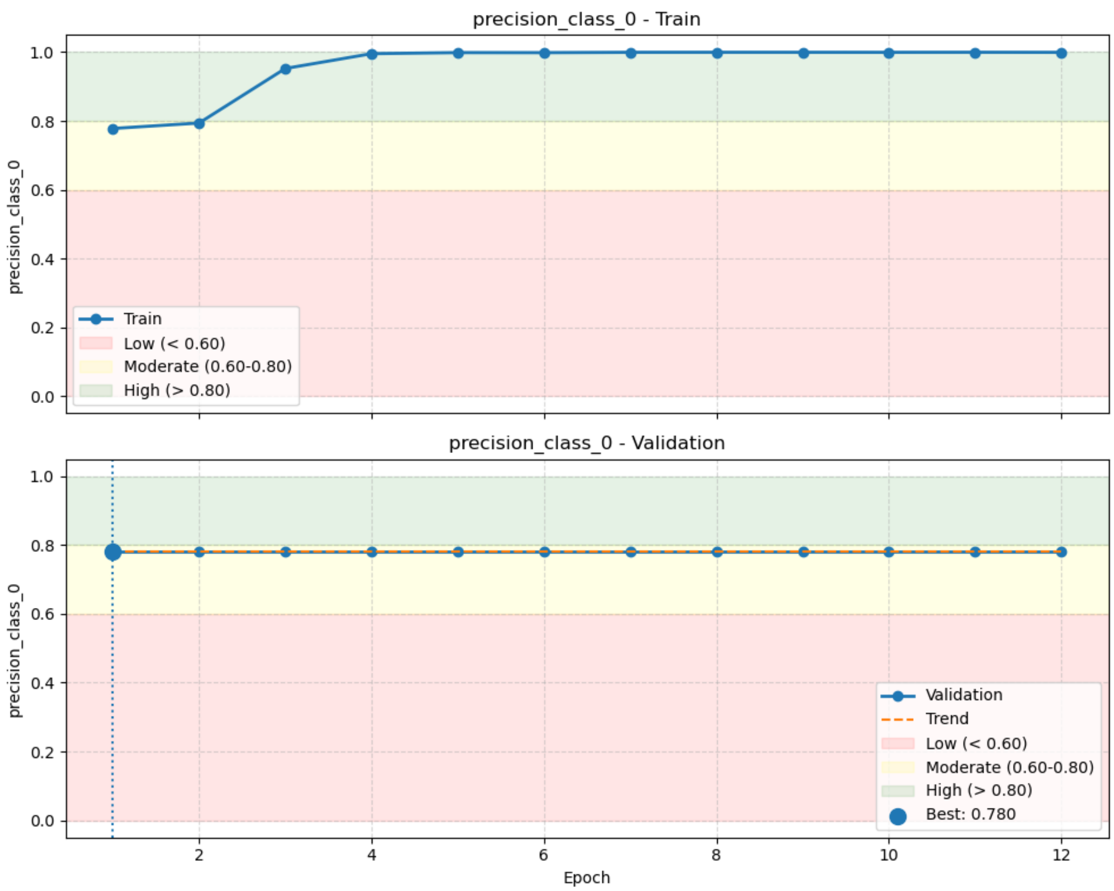
2 Per-Class Classification Metrics

2.1 Micro Aneurysms (MA)

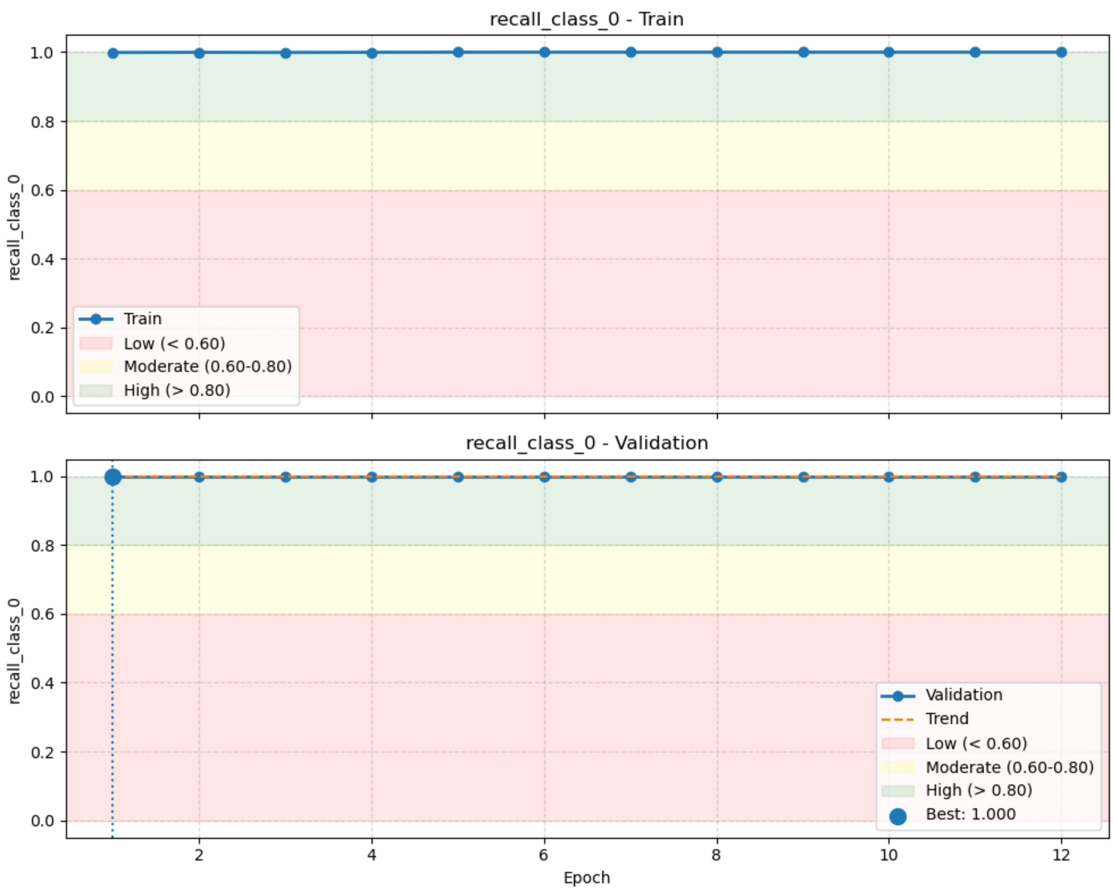
[14]:
```
plot_train_val(df, "f1_class_0")
```



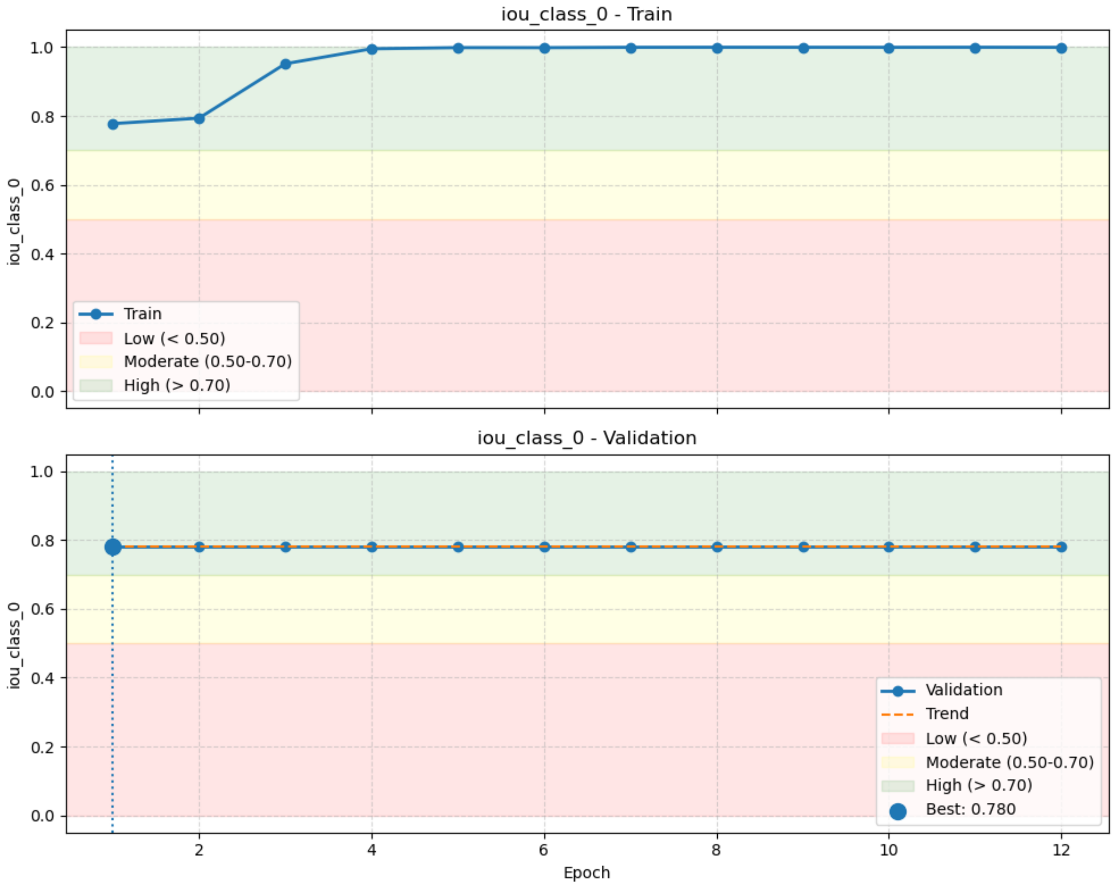

```
[15]: plot_train_val(df, "precision_class_0")
```




```
[16]: plot_train_val(df, "recall_class_0")
```

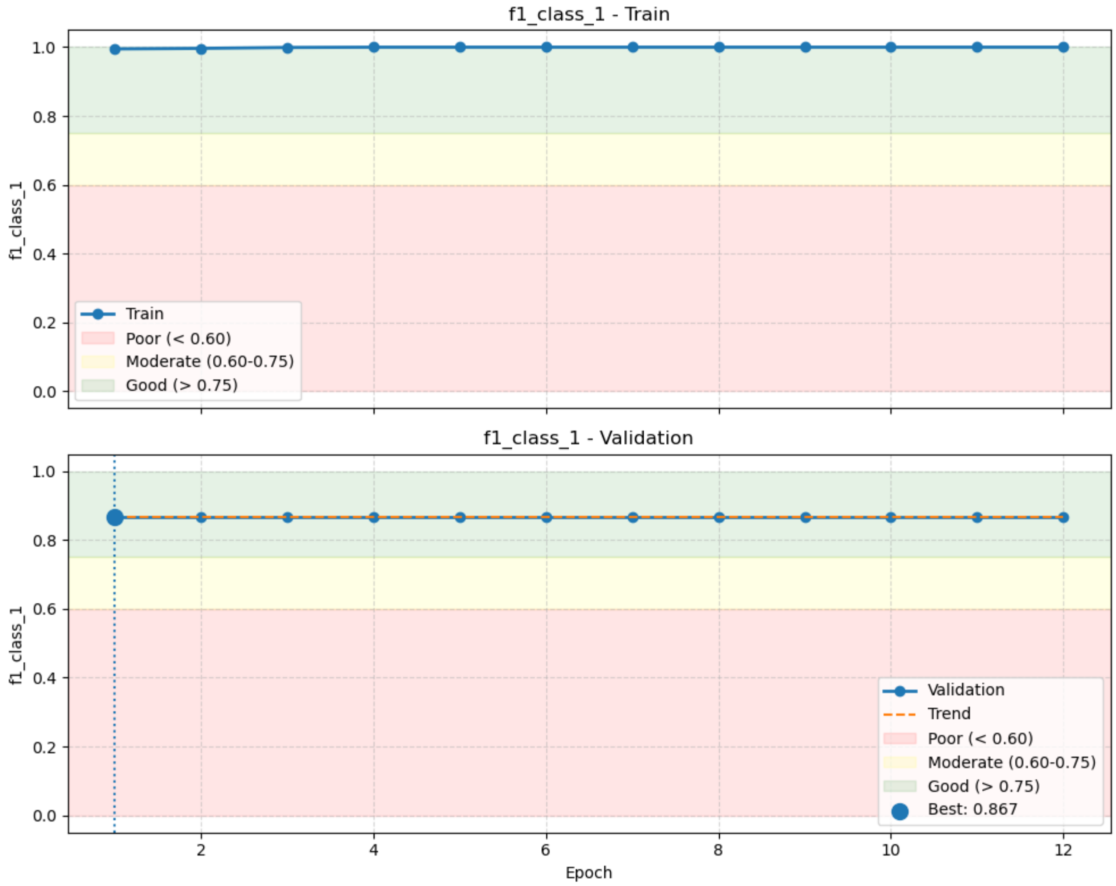


[17]:
```
plot_train_val(df, "iou_class_0")
```

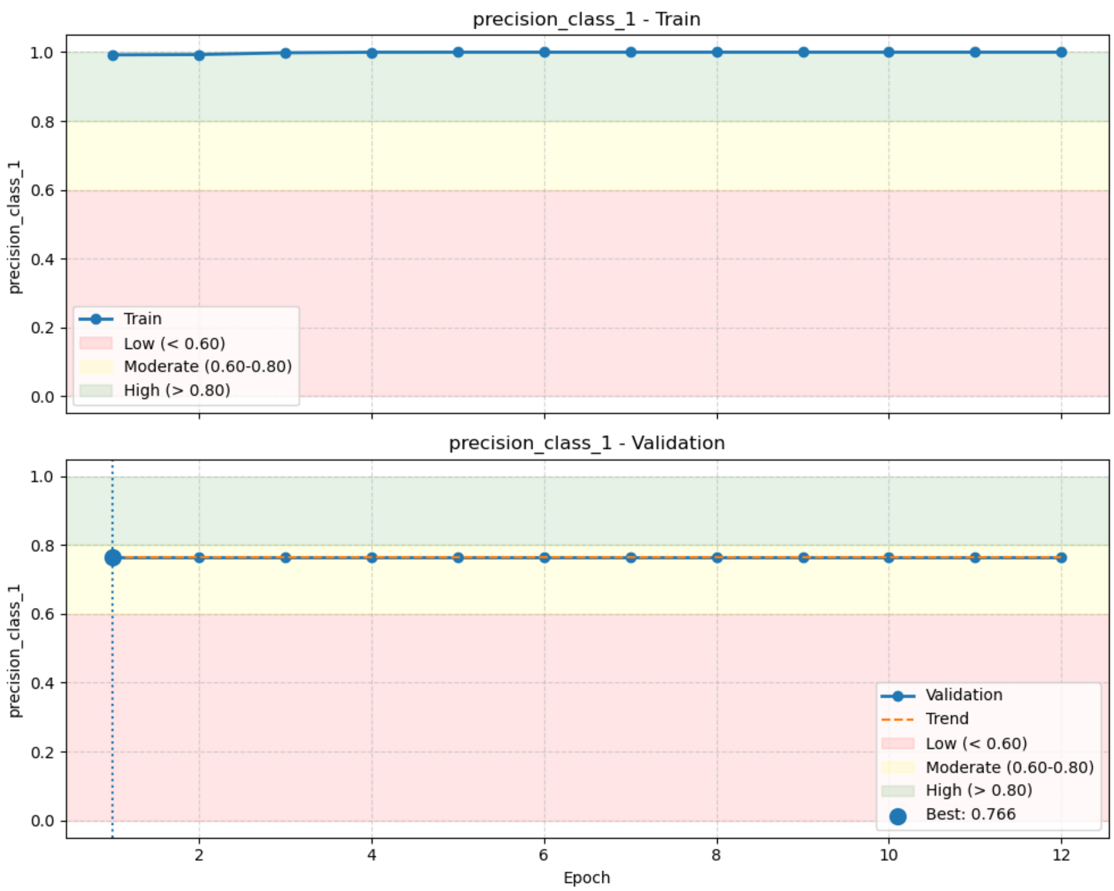


2.2 Hemorrhages (HE)

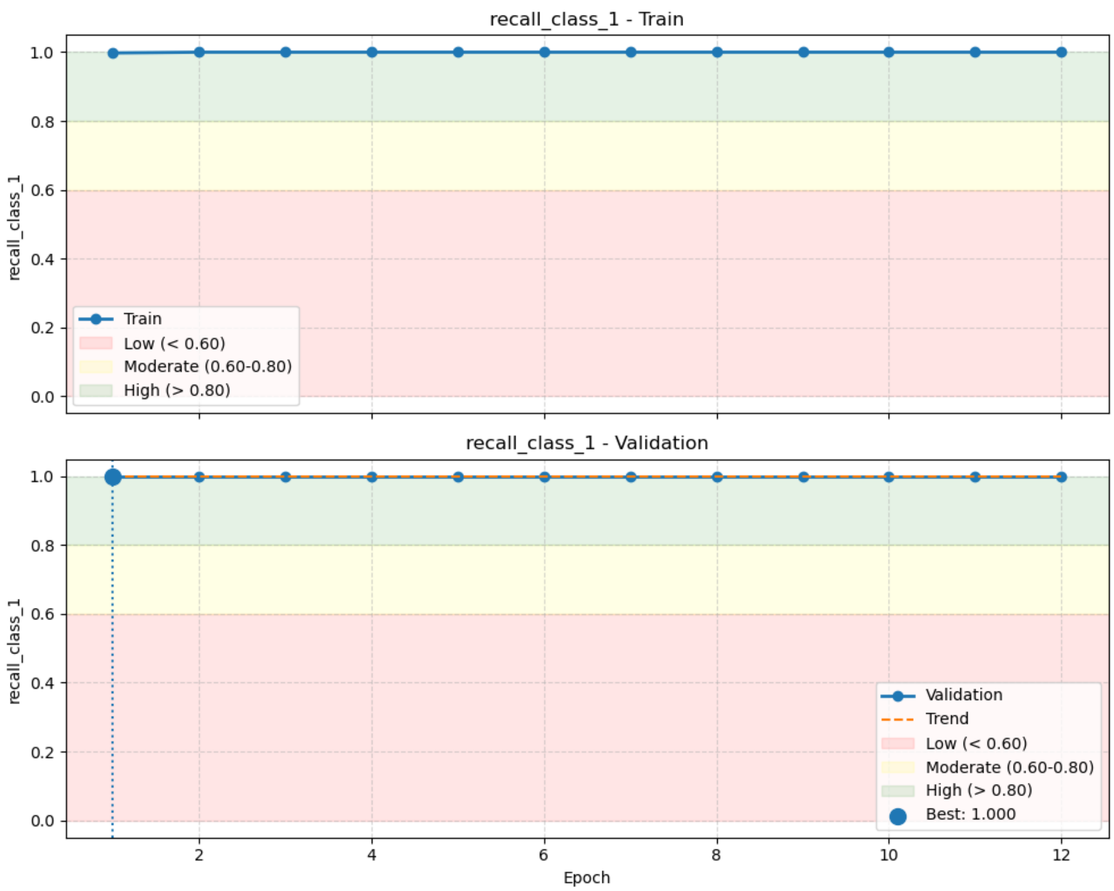
[18]:
```
plot_train_val(df, "f1_class_1")
```



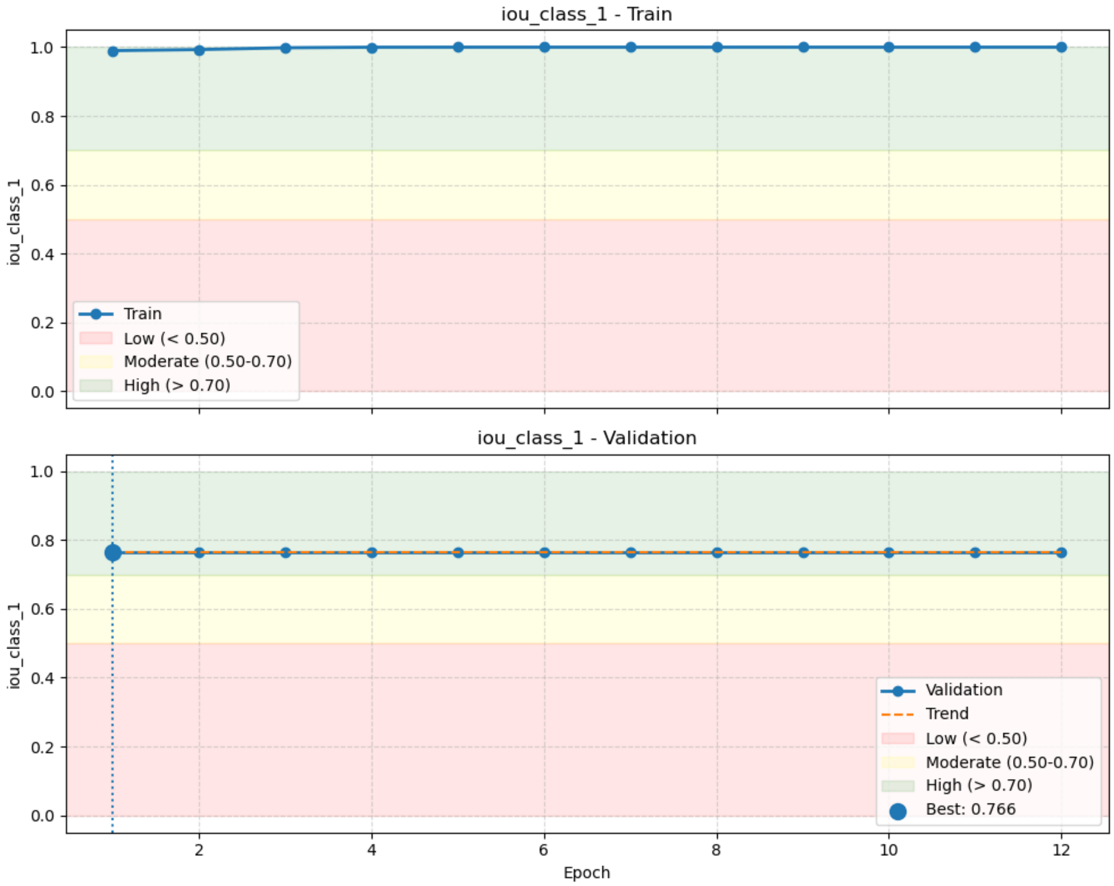
[19]:
```
plot_train_val(df, "precision_class_1")
```



[20]:
```
plot_train_val(df, "recall_class_1")
```

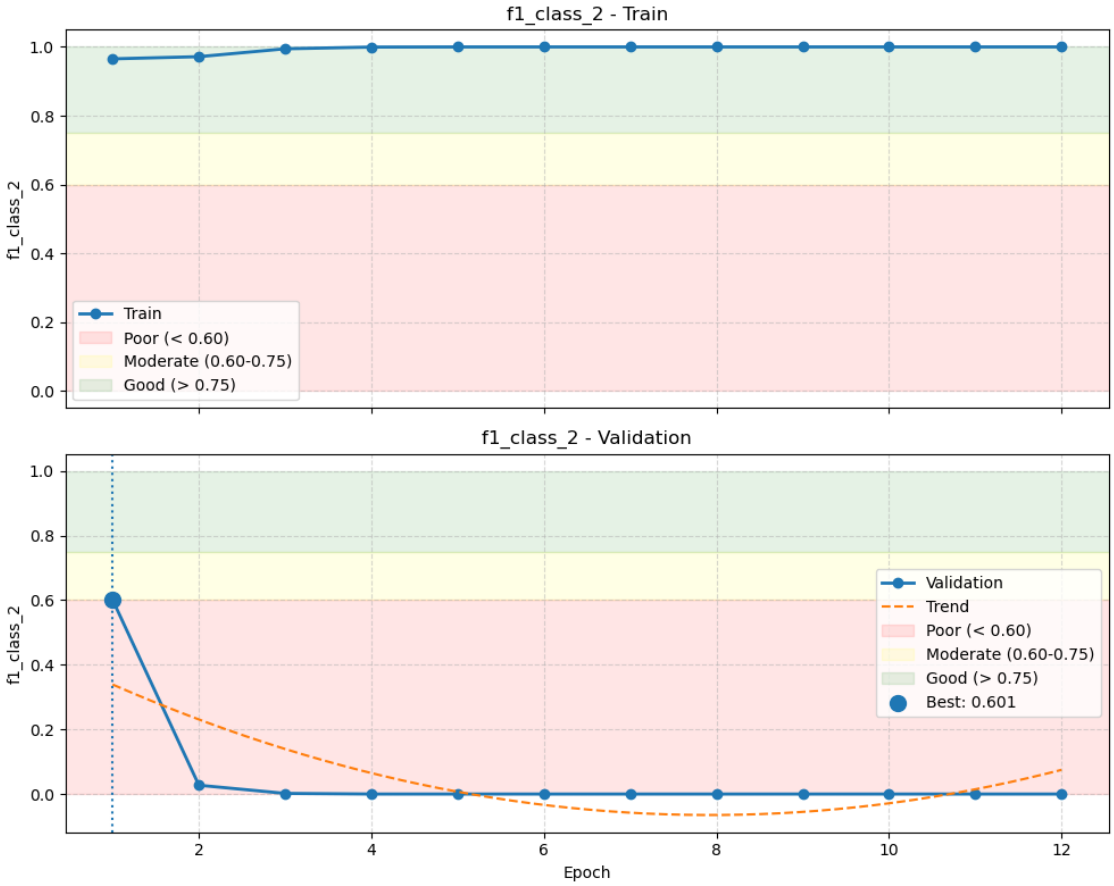


[21]:
```
plot_train_val(df, "iou_class_1")
```

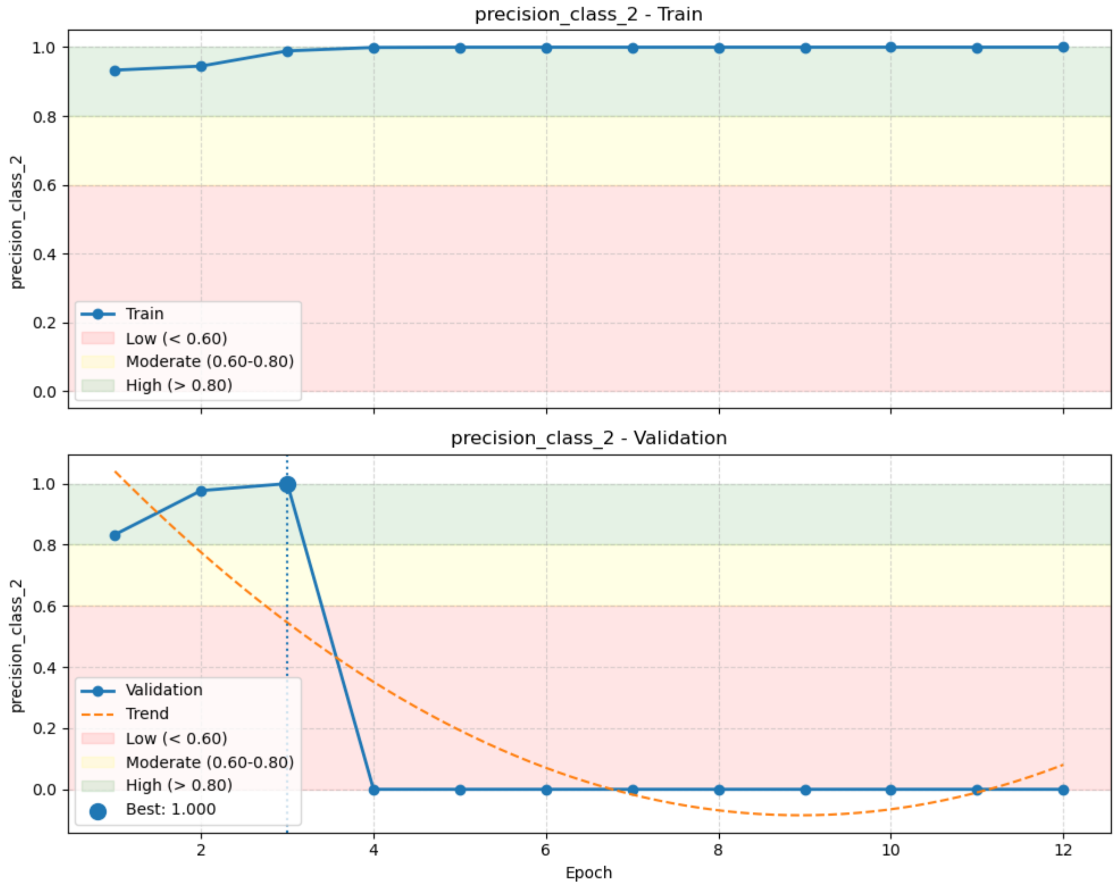


2.3 Hard Exudates (EX)

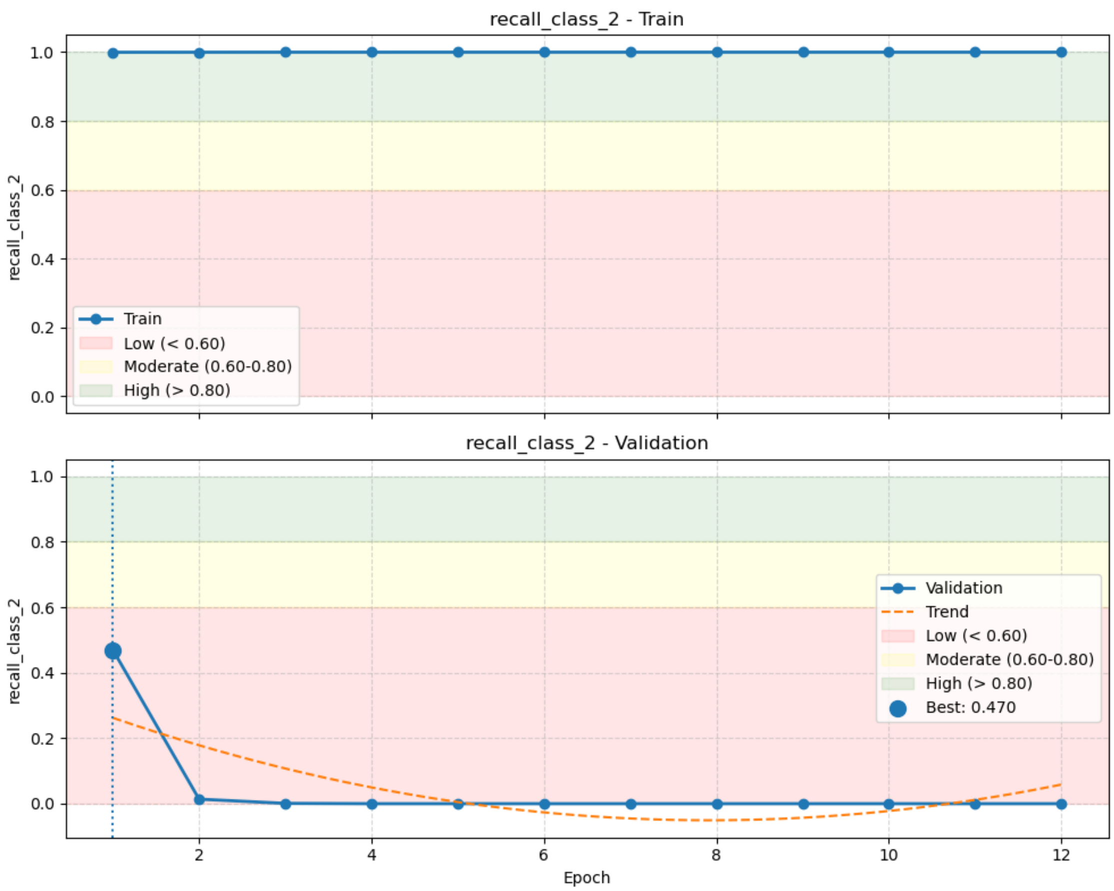
[22]:
```
plot_train_val(df, "f1_class_2")
```



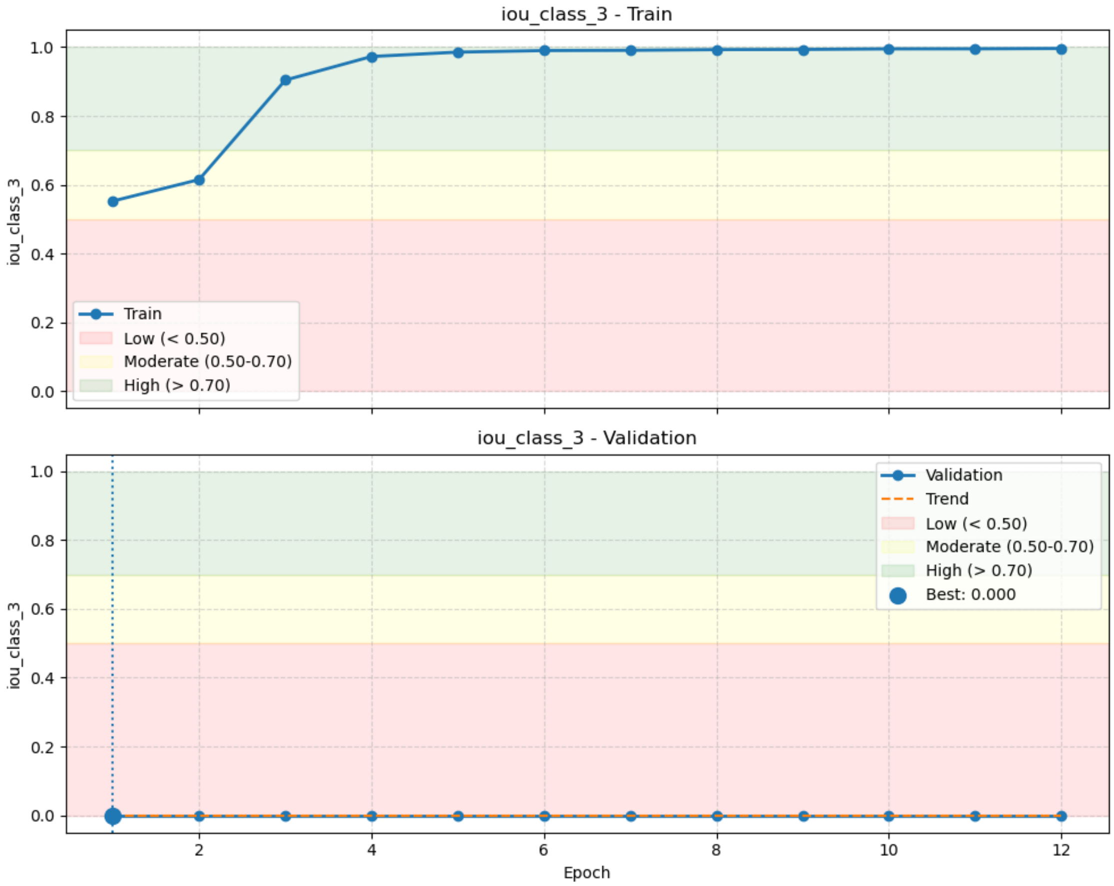
[23]:
```
plot_train_val(df, "precision_class_2")
```



[24]:
```
plot_train_val(df, "recall_class_2")
```

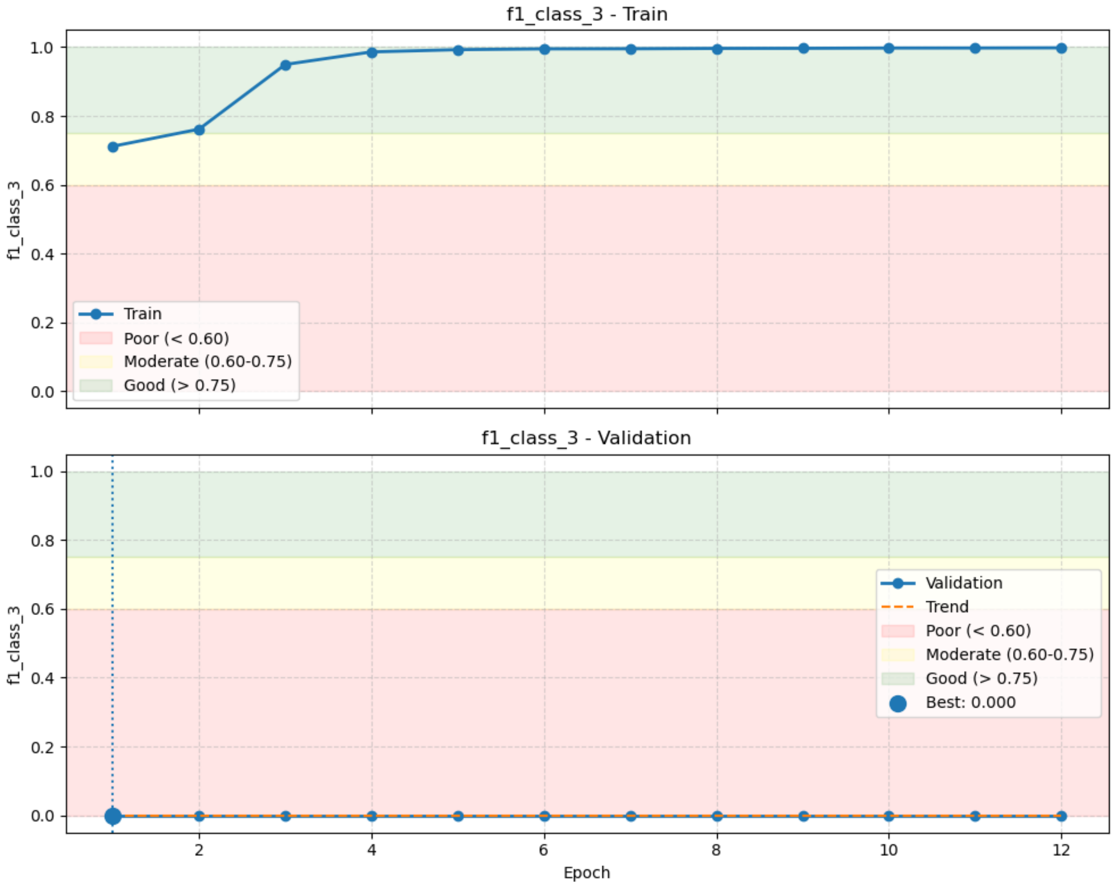



```
[25]: plot_train_val(df, "iou_class_3")
```

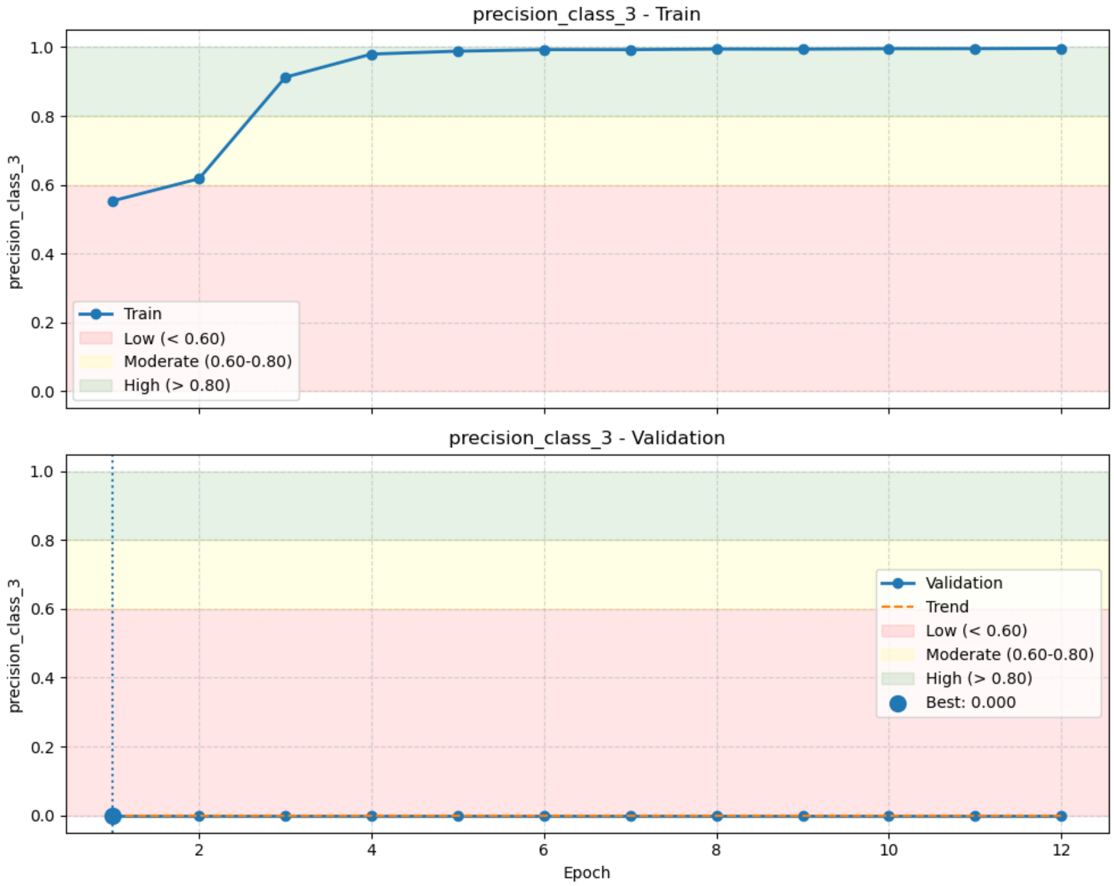


2.4 Soft Exudates (SE)

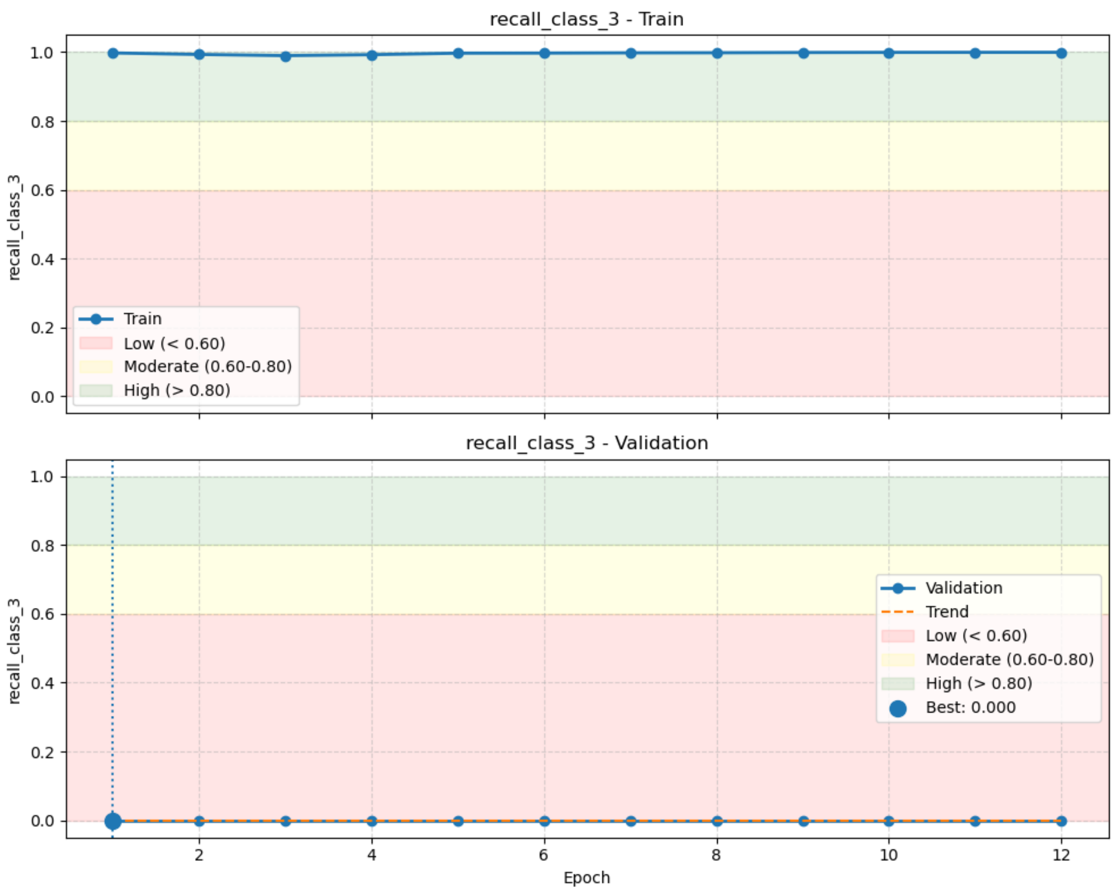
[26]:
```
plot_train_val(df, "f1_class_3")
```



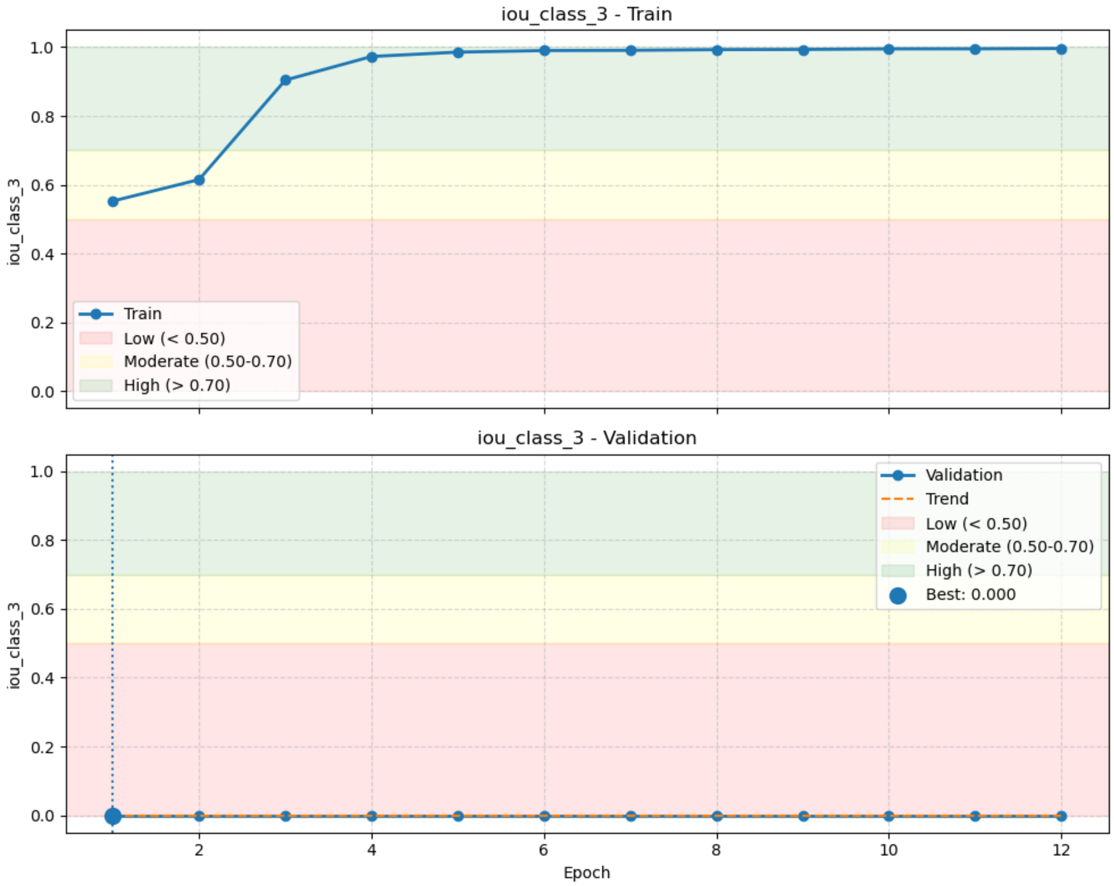
[27]:
```
plot_train_val(df, "precision_class_3")
```



[28]:
```
plot_train_val(df, "recall_class_3")
```



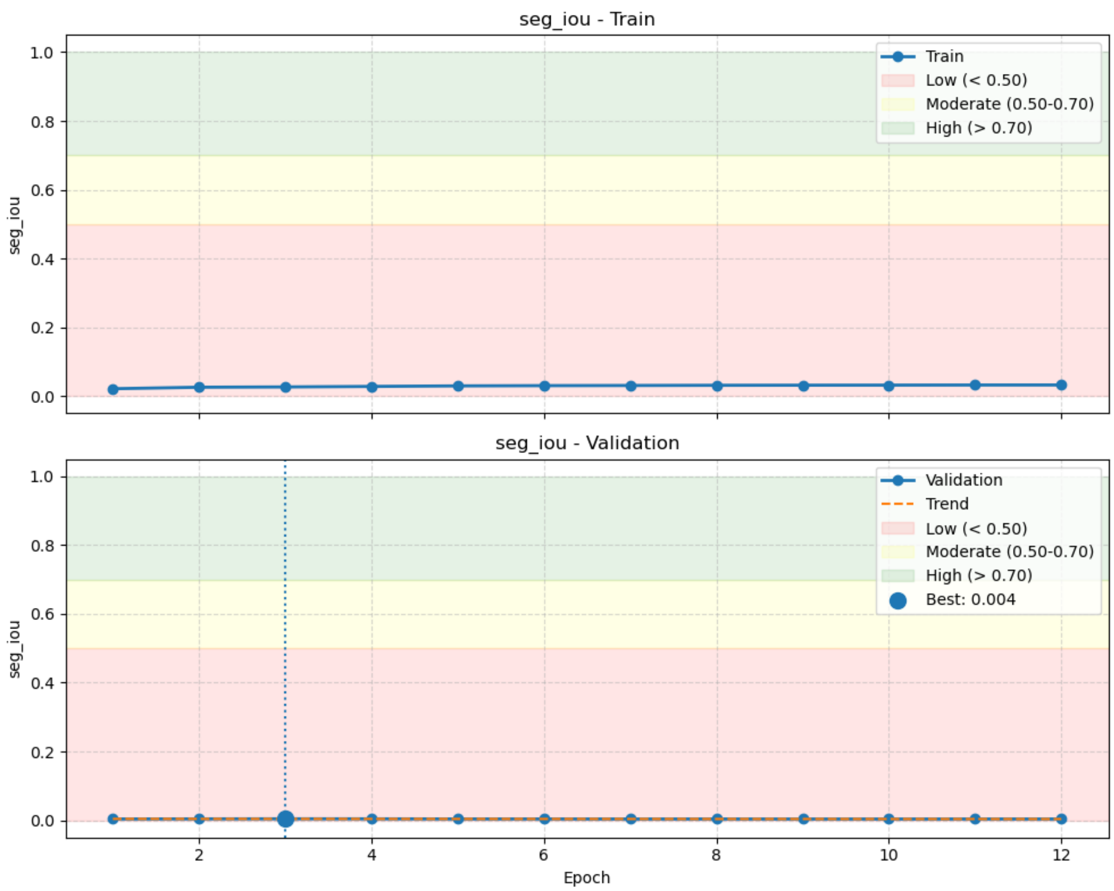
[29]:
```
plot_train_val(df, "iou_class_3")
```



2.5 Segmentation Metrics

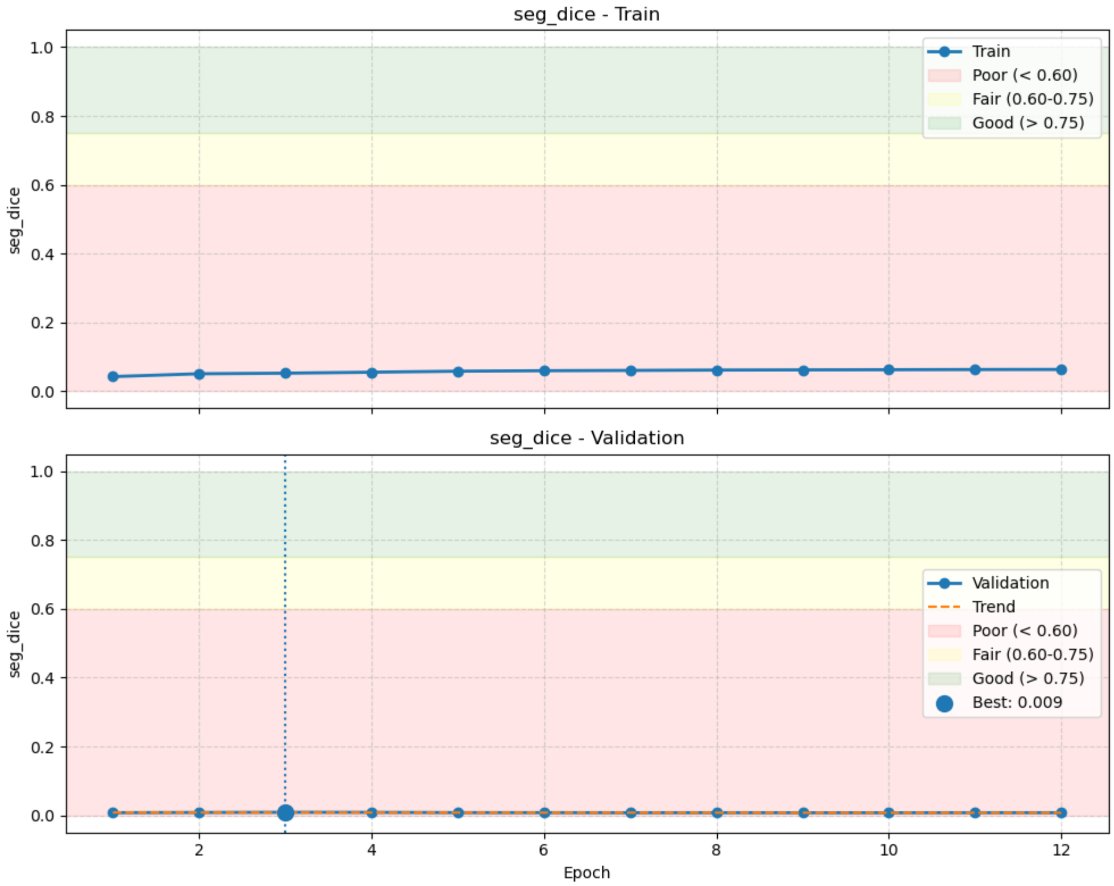
2.5.1 IoU

[30]:
```
plot_train_val(df, "seg_iou")
```



2.6 Dice

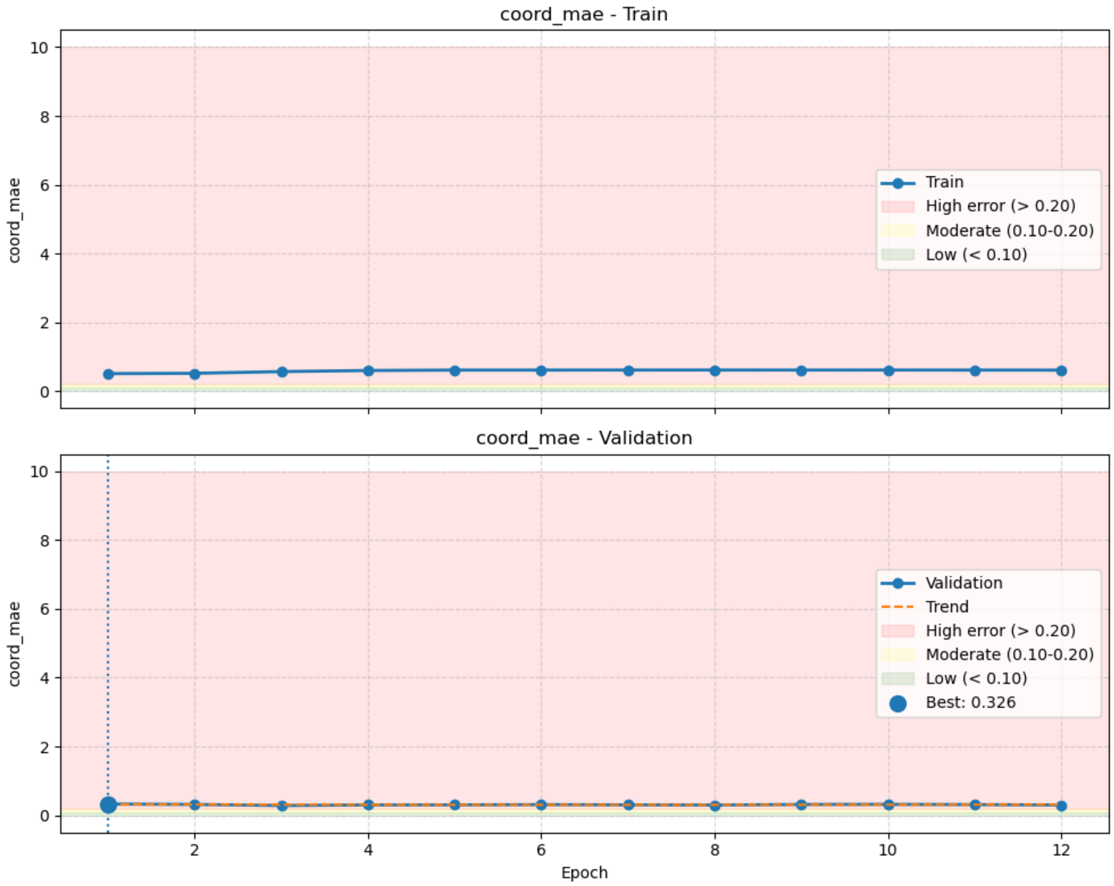
[31]:
```
plot_train_val(df, "seg_dice")
```



3 Coordinate Regression Metrics

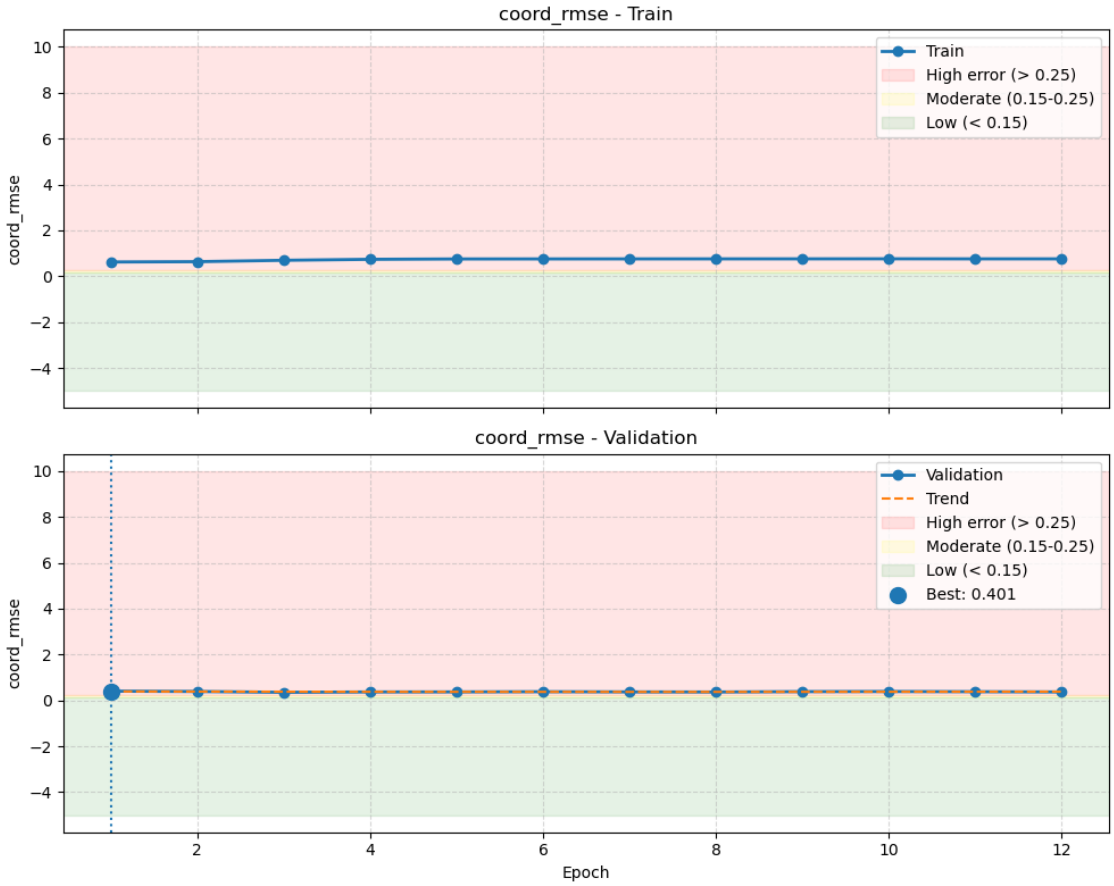
4 MAE

[32]:
```
plot_train_val(df, "coord_mae")
```



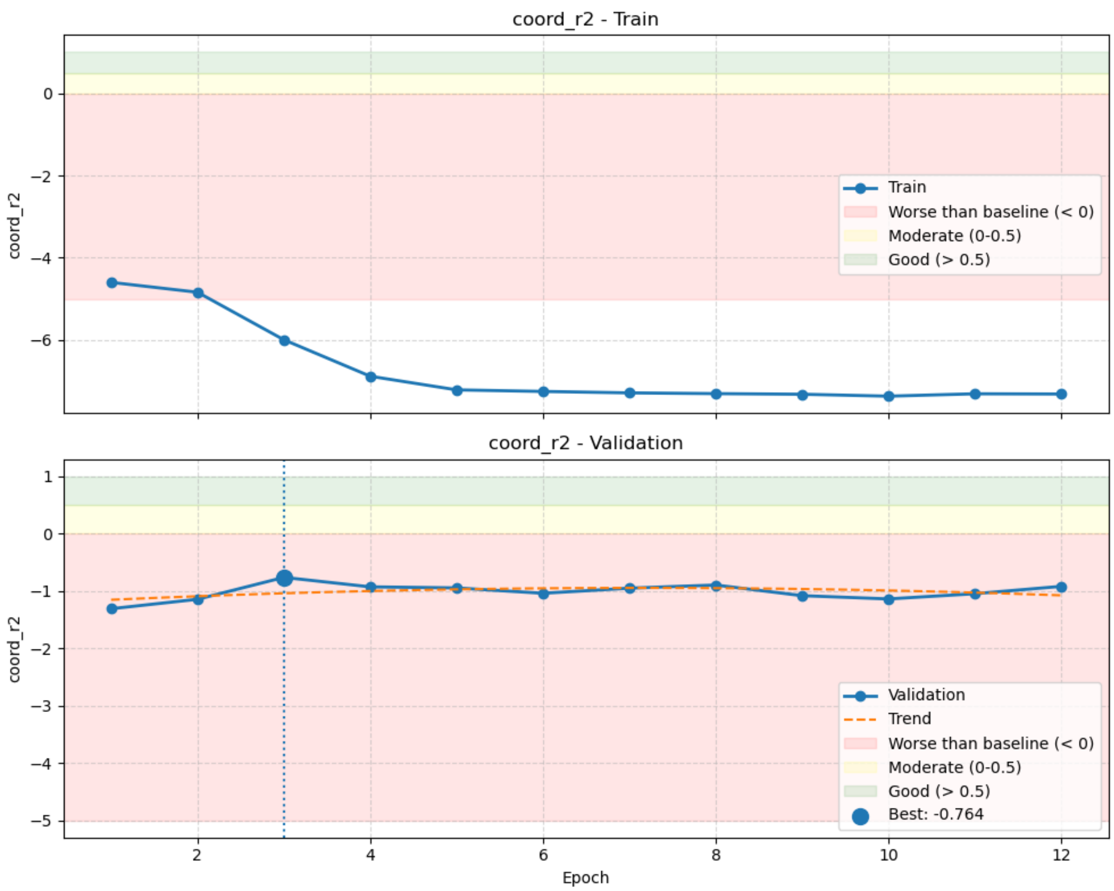
4.1 RMSE

[33]:
```
plot_train_val(df, "coord_rmse")
```



4.2 R^2

[34]:
```
plot_train_val(df, "coord_r2")
```


coord_r2 - Train
coord_r2
Train
Worse than baseline (< 0)
Moderate (0-0.5)
Good (> 0.5)
coord_r2 - Validation
coord_r2
Validation
Trend
Worse than baseline (< 0)
Moderate (0-0.5)
Good (> 0.5)
Best: -0.764
Epoch