```python
import os
import math
import numpy as np
import pandas as pd
import plotly.io as pio
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from tensorboard.backend.event_processing import event_accumulator
pio.kaleido.scope.mathjax = None
```

```python
colors = px.colors.qualitative.Plotly
print("colors:", colors)
```

```
colors: ['#636EFA', '#EF553B', '#00CC96', '#AB63FA', '#FFA15A', '#19D3F3', '#FF6692', '#B6E880', '#FF97FF', '#FECB52']
```

```python
LOG_DIR = "logs"
COLOR_MAPS = {
    "IMAXPPO": colors[1],
    "SupMAPPO": colors[3],
    "GAILMAPPO": colors[0],
    "MAPPO": colors[5],
    "IPPO": colors[8],
    "QMIX": "#feb406",
    "QPLEX": colors[2],
}
ALGOS = ["MAPPO", "IPPO", "QMIX", "QPLEX", "SupMAPPO", "GAILMAPPO", "IMAXPPO"]
ALGO_NAMES = ["MAPPO", "IPPO", "QMIX", "QPLEX", "SupMAPPO", "IMAX-PPO (GAIL)", "IMAX-PPO (InQ)"]
```

```python
def read_tensorboard_events(folder):
    logs = []
    for file in os.listdir(folder):
        if "events" in file:
            event = event_accumulator.EventAccumulator(os.path.join(folder, file))
            event.Reload()
            tag = "game_eval/winrate"
            if tag not in event.scalars.Keys():
                tag = "game/winrate"
            logs += [(x.step, x.value) for x in event.Scalars(tag)]
    logs.sort(key=lambda x: x[0])
    return logs
```

```python
def smooth(scalars, weight):
    last = 0
    smoothed = []
    for num_acc, next_val in enumerate(scalars):
        last = last * weight + (1 - weight) * next_val
        smoothed.append(last / (1 - math.pow(weight, num_acc+1)))
    return smoothed
```

```python
def read_logs(ts_factor=0.9):
    inputs = []
    for algo in ALGOS:
        for env_name in os.listdir(f"{LOG_DIR}/{algo}"):
            inputs.append((algo, env_name))
    inputs.sort(key=lambda x: ALGOS.index(x[0]))
    data = {}
    for algo, env_name in inputs:
        steps, winrates = zip(*read_tensorboard_events(f"{LOG_DIR}/{algo}/{env_name}"))
        smooth_winrates = smooth(winrates, ts_factor)
        if env_name not in data:
            data[env_name] = {}
        data[env_name][algo] = (steps, winrates, smooth_winrates)
    return data
data = read_logs()
```

```python
data_df = pd.DataFrame.from_dict(data)
data_df.to_pickle("saved_results/pickle_series.pkl")
mean_df = data_df.map(lambda x: np.mean(x[-1][-32:]) if isinstance(x, tuple) else 0)
def show_senarios(name):
    df = mean_df.filter(regex=name).copy()
    names = df.columns.tolist()
    names.sort(key=lambda x: x.replace("5_vs_5", "05_vs_05").replace("hard", "xhard"))
    df = df[names].loc[ALGOS]
    df = df.rename(index={algo: name for algo, name in zip(ALGOS, ALGO_NAMES)})
    df["Average"] = df.mean(axis=1)
    df = df.map(lambda x: f"{100*x:.1f}%")
    df = df.transpose()
    display(df)
show_senarios("protoss")
show_senarios("terran")
show_senarios("zerg")
show_senarios("miner")
show_senarios("academy")
```

	MAPPO	IPPO	QMIX	QPLEX	SupMAPPO	IMAX-PPO (GAIL)	IMAX-PPO (InQ)
protoss_5_vs_5	58.0%	54.6%	70.2%	53.3%	71.8%	68.1%	78.7%
protoss_10_vs_10	58.3%	58.0%	69.0%	53.7%	67.3%	59.6%	79.8%
protoss_10_vs_11	18.2%	20.3%	42.5%	22.8%	36.7%	21.3%	48.7%
protoss_20_vs_20	38.1%	44.5%	69.7%	27.2%	71.1%	76.3%	80.6%
protoss_20_vs_23	5.1%	4.1%	16.5%	4.8%	21.9%	11.8%	24.2%
Average	35.5%	36.3%	53.6%	32.3%	53.7%	47.4%	62.4%

	MAPPO	IPPO	QMIX	QPLEX	SupMAPPO	IMAX-PPO (GAIL)	IMAX-PPO (InQ)
terran_5_vs_5	52.0%	56.2%	58.4%	70.0%	55.8%	53.3%	69.9%
terran_10_vs_10	58.1%	57.3%	65.8%	66.1%	54.1%	58.4%	72.2%
terran_10_vs_11	28.6%	31.0%	39.4%	41.4%	26.9%	28.4%	53.9%
terran_20_vs_20	52.8%	49.6%	57.6%	23.9%	38.6%	35.9%	65.4%
terran_20_vs_23	11.2%	10.0%	10.0%	7.0%	11.2%	4.7%	17.7%
Average	40.5%	40.8%	46.2%	41.7%	37.3%	36.1%	55.8%

	MAPPO	IPPO	QMIX	QPLEX	SupMAPPO	IMAX-PPO (GAIL)	IMAX-PPO (InQ)
zerg_5_vs_5	41.0%	37.2%	37.2%	47.8%	52.5%	48.6%	55.0%
zerg_10_vs_10	39.1%	49.4%	40.8%	41.6%	57.4%	50.6%	57.6%
zerg_10_vs_11	31.2%	26.0%	28.0%	31.1%	38.1%	34.8%	41.5%
zerg_20_vs_20	31.9%	31.2%	30.4%	15.8%	44.3%	26.7%	43.3%
zerg_20_vs_23	15.8%	8.3%	10.1%	6.7%	13.6%	8.2%	21.3%
Average	31.8%	30.4%	29.3%	28.6%	41.2%	33.8%	43.7%

	MAPPO	IPPO	QMIX	QPLEX	SupMAPPO	IMAX-PPO (GAIL)	IMAX-PPO (InQ)
miner_easy_2_vs_2	48.9%	49.3%	57.2%	59.8%	47.1%	54.5%	61.8%
miner_medium_2_vs_2	40.6%	39.5%	47.3%	50.4%	39.4%	39.3%	55.0%
miner_hard_2_vs_2	31.2%	31.2%	41.7%	43.5%	31.3%	29.7%	49.8%
Average	40.2%	40.0%	48.7%	51.2%	39.2%	41.2%	55.5%

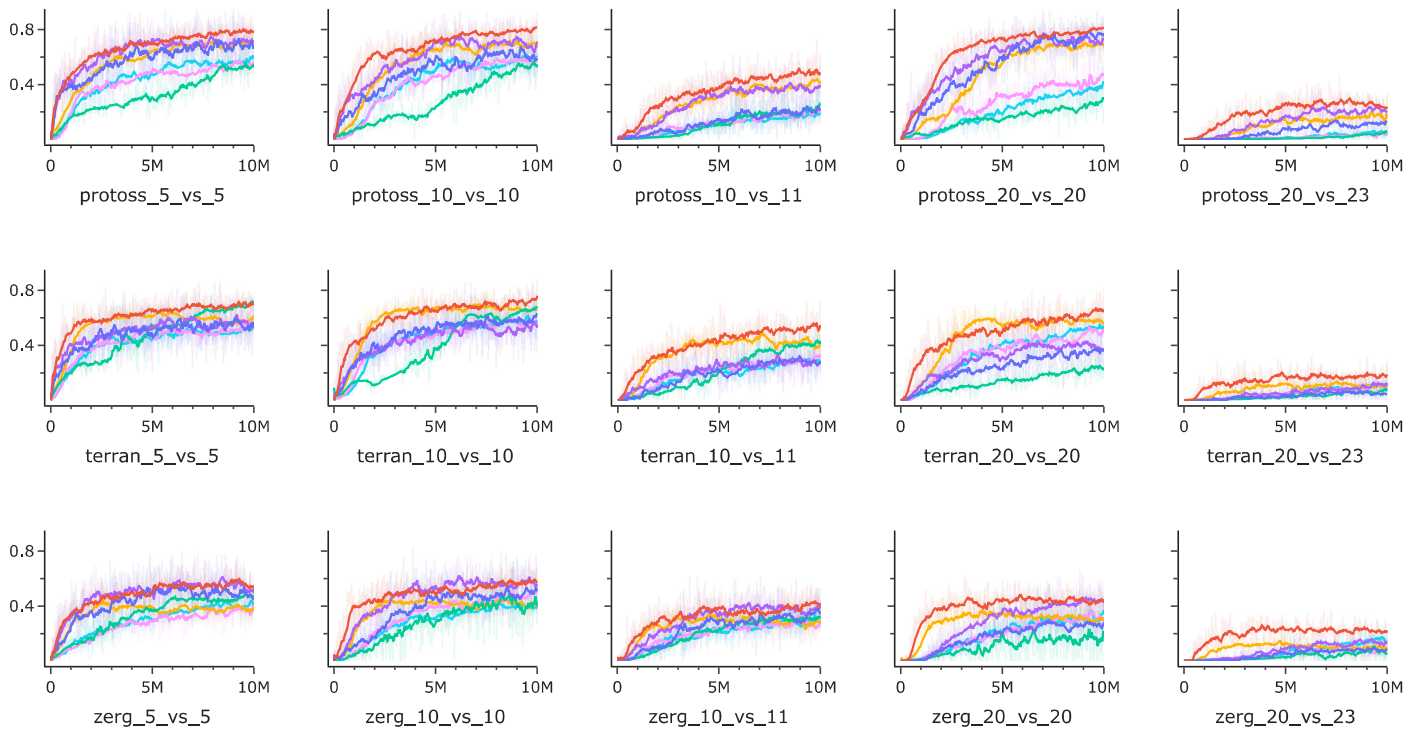
	MAPPO	IPPO	QMIX	QPLEX	SupMAPPO	IMAX-PPO (GAIL)	IMAX-PPO (InQ)
academy_3_vs_1_with_keeper	88.0%	82.7%	8.1%	90.2%	96.1%	96.4%	98.1%
academy_counterattack_easy	87.8%	84.1%	16.0%	94.9%	89.7%	64.1%	95.0%
academy_counterattack_hard	77.4%	70.9%	3.2%	95.1%	10.7%	15.2%	97.3%
Average	84.4%	79.2%	9.1%	93.4%	65.5%	58.6%	96.8%

```python
def create_baseline_fig(data, showlegend_dict, min_x=1e4, max_x=10e6, min_y=1e-3, max_y=0.95, dtick_x=5e6, dtick_y=0.4, line_width=1.6, dash="solid"):
    fig = go.Figure()
    for algo in ALGOS:
        if algo not in data:
            continue
        steps, winrates, _ = data[algo]
        fig.add_trace(go.Scatter(x=steps, y=winrates, mode='lines', showlegend=False, line_color=COLOR_MAPS[algo], opacity=0.1, line_width=1))
    for algo, name in zip(ALGOS, ALGO_NAMES):
        if algo not in data:
            continue
        steps, _, smooth_winrates = data[algo]
        showlegend = showlegend_dict.get(algo, True)
        showlegend_dict[algo] = False
        dash = "solid" if len(smooth_winrates) > 20 else dash
        fig.add_trace(go.Scatter(
            x=steps, y=smooth_winrates, mode='lines', showlegend=showlegend, name=name,
            line_color=COLOR_MAPS[algo], line_width=line_width,
            line_dash=dash
        ))
    fig.update_layout(template='simple_white', margin=dict(l=0, r=0, t=0, b=0, pad=0, autoexpand=True))
    fig.update_layout(height=140, width=220)
    if min_x is not None:
        fig.update_xaxes(range=[min_x, max_x], dtick=dtick_x, minor=dict(ticklen=2))
    if min_y is not None:
        fig.update_yaxes(range=[min_y, max_y], dtick=dtick_y, minor=dict(ticklen=3, nticks=2))
    fig.update_layout(showlegend=False)
    return fig
```

```python
plotly_figs, showlegend_dict = {}, {}
for env_name in sorted(data.keys()):
    if any(x in env_name for x in ["protoss", "terran", "zerg"]):
        plotly_figs[env_name] = create_baseline_fig(data[env_name], showlegend_dict, max_y=0.86, dtick_y=0.2)
```

```python
env_names = [env_name for env_name in plotly_figs if any(x in env_name for x in ["protoss", "terran", "zerg"])]
env_names.sort(key=lambda x: x.replace("5_vs_5", "05_vs_05"))
os.makedirs("graphs", exist_ok=True)
fig = make_subplots(rows=3, cols=5, horizontal_spacing=0.06, vertical_spacing=0.2)
for i, env_name in enumerate(env_names):
    plotly_fig = plotly_figs[env_name]
    plotly_fig.write_image(f"graphs/{env_name}.pdf")
    fig.add_traces(plotly_fig.data, rows=i//5+1, cols=i%5+1)
    fig['layout'][f'xaxis{i+1}'].update(title=env_name)
    fig['layout'][f'yaxis{i+1}'].update(showticklabels=i%5==0)
fig.update_layout(template='simple_white', margin=dict(l=4, r=4, t=4, b=4, pad=4, autoexpand=True))
fig.update_layout(height=200*3, width=180*6)
fig.update_xaxes(dtick=5e6, minor=dict(ticklen=2))
fig.update_yaxes(range=[1e-3, 0.95], dtick=0.4, minor=dict(ticklen=3, nticks=2))
fig.update_layout(legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1))
fig.update_layout(autosize=False)
fig.show("svg")
fig.update_layout(height=500, width=1000)
fig.update_layout(legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1))
fig.write_image(f"graphs/legend.pdf")
```

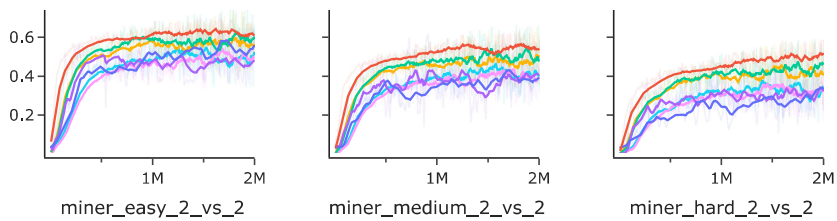
Legend: MAPPO — IPPO — QMIX — QPLEX — SupMAPPO — IMAX-PPO (GAIL) — IMAX-PPO (InQ)

Top grid (x-axis 0 to 10M, y-axis marked 0.4, 0.8):
Row 1: protoss_5_vs_5, protoss_10_vs_10, protoss_10_vs_11, protoss_20_vs_20, protoss_20_vs_23
Row 2: terran_5_vs_5, terran_10_vs_10, terran_10_vs_11, terran_20_vs_20, terran_20_vs_23
Row 3: zerg_5_vs_5, zerg_10_vs_10, zerg_10_vs_11, zerg_20_vs_20, zerg_20_vs_23

```
In [ ]:  plotly_figs, showlegend_dict = {}, {}
         for env_name in sorted(data.keys()):
             if "miner" in env_name:
                 plotly_figs[env_name] = create_baseline_fig(data[env_name], showlegend_dict, max_x=2e6, max_y=0.69, min_y=0.01, dtick_x=1e6, dtick_y=0.2)
```

```
In [ ]:  env_names = [env_name for env_name in plotly_figs if "miner" in env_name]
         print("env_names:", env_names)
         env_names.sort(key=lambda x: x.replace("hard", "xhard"))
         os.makedirs("graphs", exist_ok=True)
         fig = make_subplots(rows=1, cols=5, horizontal_spacing=0.06, vertical_spacing=0.2)
         for i, env_name in enumerate(env_names):
             plotly_fig = plotly_figs[env_name]
             plotly_fig.write_image(f"graphs/{env_name}.pdf")
             fig.add_traces(plotly_fig.data, rows=i//5+1, cols=i%5+1)
             fig['layout'][f'xaxis{i+1}'].update(title=env_name)
             fig['layout'][f'yaxis{i+1}'].update(showticklabels=i%5==0)
         fig.update_layout(template='simple_white', margin=dict(l=4, r=4, t=4, b=4, pad=4, autoexpand=True))
         fig.update_layout(height=200, width=180*6)
         fig.update_xaxes(dtick=1e6, minor=dict(ticklen=3,nticks=2))
         fig.update_yaxes(range=[0.01, 0.74], dtick=0.2, minor=dict(ticklen=3, nticks=1))
         fig.update_layout(legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1))
         fig.update_layout(autosize=False)
         fig.show("svg")
```

env_names: ['miner_easy_2_vs_2', 'miner_hard_2_vs_2', 'miner_medium_2_vs_2']

Legend: MAPPO — IPPO — QMIX — QPLEX — SupMAPPO — IMAX-PPO (GAIL) — IMAX-PPO (InQ)

Middle grid (x-axis 0 to 2M, y-axis marked 0.2, 0.4, 0.6):
miner_easy_2_vs_2, miner_medium_2_vs_2, miner_hard_2_vs_2

```
In [ ]:  plotly_figs, showlegend_dict = {}, {}
         for env_name in sorted(data.keys()):
             if "academy" in env_name:
                 plotly_figs[env_name] = create_baseline_fig(data[env_name], showlegend_dict, max_x=5e5, dtick_x=2e5, max_y=1.1, dtick_y=0.5, dash="dash")
```

```
In [ ]:  env_names = [env_name for env_name in plotly_figs if "academy" in env_name]
         print("env_names:", env_names)
         env_names.sort(key=lambda x: x.replace("hard", "xhard"))
         os.makedirs("graphs", exist_ok=True)
         fig = make_subplots(rows=1, cols=5, horizontal_spacing=0.06, vertical_spacing=0.2)
         for i, env_name in enumerate(env_names):
             plotly_fig = plotly_figs[env_name]
             plotly_fig.write_image(f"graphs/{env_name}.pdf")
             fig.add_traces(plotly_fig.data, rows=i//5+1, cols=i%5+1)
             fig['layout'][f'xaxis{i+1}'].update(title=env_name)
             fig['layout'][f'yaxis{i+1}'].update(showticklabels=i%5==0)
         fig.update_layout(template='simple_white', margin=dict(l=4, r=4, t=4, b=4, pad=4, autoexpand=True))
         fig.update_layout(height=200, width=180*6)
         fig.update_xaxes(range=[1e2, 5e5], dtick=2e5, minor=dict(ticklen=3, nticks=2))
         fig.update_layout(legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1))
         fig.update_layout(autosize=False)
         fig.show("svg")
         fig.update_layout(height=500, width=1000)
         fig.update_layout(legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1))
         fig.write_image(f"graphs/legend_grf.pdf")
```

env_names: ['academy_3_vs_1_with_keeper', 'academy_counterattack_easy', 'academy_counterattack_hard']

academy_3_vs_1_with_keeper   academy_counterattack_easy   academy_counterattack_hard