

```
In [ ]: import os
import pickle
import pandas as pd
import plotly.io as pio
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from trainer.utils import read_logs, smooth
pio.kaleido.scope.mathjax = None
pd.options.plotting.backend = "plotly"

In [ ]: LOG_DIR = "logs"
COLOR_MAPS = {
    "GAIL": "#2385ca",
    "AIRL": "#23cabcc",
    "SQIL": "#078f48",
    "IIQ": "#996515",
    "MIFQ-SOFT": "#d62728",
    "MIFQ-DQN": "#d66cf27",
    "MIFQ-DQN-SIGMOID": "#2385ca",
    "MIFQ-DQN-TANH": "#23cabcc",
    "MIFQ-SOFT-SIGMOID": "#078f48",
    "MIFQ-SOFT-TANH": "#996515",
    "BC": "#febf2b",
    "IQVDN": "#62a8d8",
}
ALGOS = ["BC", "IIQ", "IQVDN", "SQIL", "AIRL", "GAIL", "MIFQ-DQN", "MIFQ-SOFT", "MIFQ-DQN-SIGMOID", "MIFQ-SOFT-SIGMOID", "MIFQ-DQN-TANH", "MIFQ-SOFT-TANH"]
ALGO_NAMES = {
    "GAIL": "MAGAIL",
    "AIRL": "MAAIRL",
    "SQIL": "MASQIL",
    "IIQ": "IIQ",
    "MIFQ-SOFT": "MIFQ (Soft)",
    "MIFQ-DQN": "MIFQ (Det)",
    "MIFQ-DQN-SIGMOID": "MIFQ (Det-Sigmoid)",
    "MIFQ-DQN-TANH": "MIFQ (Det-Tanh)",
    "MIFQ-SOFT-SIGMOID": "MIFQ (Soft-Sigmoid)",
    "MIFQ-SOFT-TANH": "MIFQ (Soft-Tanh)",
    "BC": "BC",
    "Expert": "Expert",
    "IQVDN": "IQVDN",
}

In [ ]: data = read_logs(LOG_DIR)

In [ ]: os.makedirs("saved_results", exist_ok=True)
with open("saved_results/log_data.pkl", "wb") as f:
    pickle.dump(data, f)

In [ ]: all_eps = sorted([int(x.replace("eps", "")) for x in data])

In [ ]: def show_table(task="smac", eps=4096, percentage=True, std=False):
    # for eps in all_eps:
    data_df = {}
    values = data[f"eps{eps}"]
    env_names = sorted(values.keys(), key=lambda x: x.replace("5_vs_5", "05_vs_05").replace("hard", "xhard").replace("speaker", "xspeaker"))
    for env_name in env_names:
        if task != "smac":
            if not env_name.startswith(task):
                continue
        elif not any(x in env_name for x in ["protoss", "terran", "zerg"]):
            continue
        algo_values = values[env_name]
        data_df[env_name] = {}
        for algo in ALGOS:
            if algo not in algo_values:
                continue
            algo_name = ALGO_NAMES[algo]
            infos = algo_values[algo]
            if "expert_winrate" in infos and "Expert" not in data_df[env_name]:
                if std:
                    winrate = infos["expert_winrate"]
                    if percentage:
                        winrate = winrate * 100
                    prefix = "%" if percentage > 0 else ""
                    data_df[env_name]["Expert"] = f'{winrate:.1f}{prefix}'
                else:
                    data_df[env_name]["Expert"] = infos["expert_winrate"]
            if std:
                winrate = infos["winrate"]
                winrate_std = infos["winrate_std"]
                if percentage:
                    winrate = winrate * 100
                    winrate_std = winrate_std * 100
                prefix = "%" if percentage > 0 else ""
                data_df[env_name][algo_name] = f'{winrate:.1f}{prefix}±{winrate_std:.1f}{prefix}'
            else:
                data_df[env_name][algo_name] = infos["winrate"]
        if not data_df:
            return
    df = pd.DataFrame.from_dict(data_df)
    df = df.rename(index=ALGO_NAMES)
    if not std:
        df["Average"] = df.mean(axis=1)
        if percentage:
            df = df.map(lambda x: f"{100*x:.1f}%")
        else:
            df = df.map(lambda x: f"{x:.1f}")
    df = df.transpose()
    print("eps:", eps)
    display(df)

In [ ]: for eps in all_eps:
    show_table(task="smac", eps=eps, std=True)
    show_table(task="miner", eps=eps, std=True)
```

eps: 128

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
protoss_5_vs_5	86.7%	17.2%±4.7%	13.3%±2.6%	8.6%±2.6%	0.8%±1.4%	0.8%±0.6%	0.0%±0.0%	18.0%±4.6%	41.4%±4.6%
protoss_10_vs_10	90.3%	6.2%±4.4%	2.3%±1.4%	3.1%±3.8%	0.0%±0.0%	0.0%±0.0%	0.0%±0.0%	17.2%±5.2%	32.8%±8.4%
terran_5_vs_5	81.7%	9.4%±4.9%	12.5%±3.1%	9.4%±9.1%	0.0%±0.0%	0.0%±0.0%	0.0%±0.0%	25.0%±4.4%	32.0%±3.4%
terran_10_vs_10	81.7%	3.9%±3.4%	1.6%±2.7%	0.0%±0.0%	0.0%±0.0%	0.0%±0.0%	0.0%±0.0%	3.1%±2.2%	35.2%±7.1%
zerg_5_vs_5	73.5%	7.0%±4.6%	10.2%±5.6%	4.2%±4.2%	3.9%±2.6%	2.0%±1.2%	3.1%±3.8%	5.5%±2.6%	24.2%±4.1%
zerg_10_vs_10	76.3%	10.2%±3.4%	0.8%±1.4%	0.0%±0.0%	0.0%±0.0%	2.1%±2.4%	0.0%±0.0%	10.9%±1.6%	28.9%±9.2%

eps: 128

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
miner_easy_2_vs_2	82.4%	28.9%±8.1%	10.2%±3.4%	13.3%±1.4%	12.1%±2.0%	14.1%±4.7%	10.5%±3.7%	14.1%±8.4%	29.7%±6.4%
miner_medium_2_vs_2	74.9%	23.4%±3.5%	7.0%±3.4%	9.4%±3.8%	11.5%±3.5%	13.3%±1.4%	10.0%±2.4%	11.3%±1.8%	21.1%±8.1%
miner_hard_2_vs_2	69.8%	15.6%±5.8%	5.5%±1.4%	7.0%±1.4%	7.4%±1.6%	4.7%±2.7%	7.0%±1.5%	8.0%±3.2%	14.8%±4.1%

eps: 256

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
protoss_5_vs_5	86.7%	12.5%±2.2%	16.5%±2.4%	15.1%±1.6%	1.6%±1.6%	7.2%±3.2%	14.1%±4.7%	40.6%±10.4%	53.9%±6.8%
protoss_10_vs_10	90.3%	6.2%±4.9%	3.9%±2.6%	2.9%±3.4%	0.0%±0.0%	0.2%±0.3%	15.0%±4.2%	32.3%±2.9%	59.4%±8.6%
terran_5_vs_5	81.7%	10.9%±4.7%	12.5%±2.2%	14.8%±2.6%	0.8%±1.4%	0.0%±0.0%	0.0%±0.0%	27.3%±5.6%	46.1%±7.5%
terran_10_vs_10	81.7%	4.7%±1.6%	3.9%±4.1%	0.0%±0.0%	0.0%±0.0%	0.0%±0.0%	0.0%±0.0%	0.0%±0.0%	43.8%±16.7%
zerg_5_vs_5	73.5%	8.6%±2.6%	11.7%±2.6%	7.8%±6.0%	1.6%±1.6%	2.9%±2.2%	6.2%±2.2%	12.5%±3.8%	35.9%±5.2%
zerg_10_vs_10	76.3%	5.5%±2.6%	2.3%±1.4%	0.0%±0.0%	2.3%±2.6%	0.4%±0.4%	3.1%±2.2%	12.5%±3.8%	41.4%±3.4%

eps: 256

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
miner_easy_2_vs_2	82.4%	38.3%±6.0%	18.2%±6.2%	20.3%±5.6%	26.2%±2.8%	14.8%±4.6%	19.9%±2.5%	26.6%±6.4%	34.4%±4.4%
miner_medium_2_vs_2	74.9%	21.9%±10.8%	10.9%±6.4%	10.2%±2.6%	15.0%±4.0%	12.5%±3.8%	17.0%±3.2%	19.9%±3.4%	32.8%±8.4%
miner_hard_2_vs_2	69.8%	20.3%±1.6%	6.2%±1.9%	7.8%±3.5%	13.5%±3.4%	8.6%±4.1%	10.4%±2.2%	7.8%±1.7%	17.2%±8.1%

eps: 512

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
protoss_5_vs_5	86.7%	21.9%±6.6%	17.2%±1.6%	19.5%±8.1%	0.0%±0.0%	4.9%±1.4%	10.2%±5.1%	43.8%±3.1%	62.5%±7.7%
protoss_10_vs_10	90.3%	5.5%±2.6%	7.0%±2.6%	9.1%±2.3%	0.0%±0.0%	0.0%±0.0%	18.0%±7.1%	35.9%±11.4%	62.5%±9.4%
terran_5_vs_5	81.7%	9.4%±4.9%	18.0%±3.4%	24.2%±6.8%	4.7%±3.5%	0.8%±1.0%	3.9%±4.1%	39.1%±10.0%	59.4%±8.6%
terran_10_vs_10	81.7%	7.0%±4.1%	7.8%±6.4%	18.8%±18.8%	1.6%±1.6%	0.0%±0.0%	0.0%±0.0%	18.0%±7.8%	51.6%±7.8%
zerg_5_vs_5	73.5%	6.2%±5.8%	10.9%±3.5%	11.7%±8.4%	17.2%±3.5%	2.9%±0.3%	15.6%±3.1%	33.6%±7.8%	51.6%±9.2%
zerg_10_vs_10	76.3%	5.5%±4.1%	3.1%±3.8%	0.0%±0.0%	0.8%±1.4%	0.2%±0.3%	1.6%±1.6%	35.2%±4.1%	46.1%±4.1%

eps: 512

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
miner_easy_2_vs_2	82.4%	30.5%±5.1%	13.9%±2.9%	16.4%±6.0%	25.8%±4.7%	20.3%±3.5%	25.0%±4.3%	31.2%±3.1%	53.9%±4.1%
miner_medium_2_vs_2	74.9%	18.8%±3.1%	14.3%±3.4%	12.5%±2.2%	19.5%±5.6%	11.7%±3.4%	20.3%±4.7%	19.9%±3.7%	39.1%±3.5%
miner_hard_2_vs_2	69.8%	15.6%±5.8%	6.1%±0.9%	5.5%±2.6%	14.1%±2.4%	7.0%±2.6%	12.5%±1.8%	14.1%±4.6%	30.5%±4.1%

eps: 1024

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
protoss_5_vs_5	86.7%	16.4%±8.9%	23.4%±3.5%	25.8%±3.4%	3.9%±3.4%	24.4%±2.5%	43.8%±6.6%	49.2%±5.6%	75.8%±2.6%
protoss_10_vs_10	90.3%	4.7%±4.7%	16.4%±4.1%	29.2%±9.3%	0.0%±0.0%	3.9%±3.4%	3.9%±6.8%	53.1%±5.8%	71.1%±2.6%
terran_5_vs_5	81.7%	12.5%±5.4%	27.3%±3.4%	33.6%±10.2%	0.0%±0.0%	1.0%±0.6%	0.8%±1.4%	52.3%±2.6%	60.9%±6.8%
terran_10_vs_10	81.7%	7.8%±8.1%	14.1%±1.6%	13.3%±1.4%	2.3%±1.4%	0.0%±0.0%	0.0%±0.0%	0.0%±0.0%	58.6%±2.6%
zerg_5_vs_5	73.5%	5.5%±4.6%	9.4%±2.2%	13.5%±1.8%	19.5%±3.4%	4.9%±4.4%	32.0%±10.5%	38.3%±5.1%	58.6%±3.4%
zerg_10_vs_10	76.3%	9.4%±5.4%	11.7%±3.4%	0.0%±0.0%	1.6%±1.6%	2.3%±2.6%	0.8%±1.4%	41.4%±5.6%	52.3%±7.5%

eps: 1024

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
miner_easy_2_vs_2	82.4%	34.4%±8.6%	17.8%±2.3%	15.6%±5.4%	35.9%±4.2%	22.7%±3.4%	31.1%±3.0%	36.9%±3.1%	54.7%±8.1%
miner_medium_2_vs_2	74.9%	20.3%±4.7%	13.3%±4.1%	10.9%±4.7%	23.2%±1.9%	14.1%±1.6%	22.1%±2.4%	36.7%±8.7%	42.2%±1.6%
miner_hard_2_vs_2	69.8%	18.0%±2.6%	9.0%±2.3%	8.6%±4.1%	19.7%±4.0%	8.6%±2.6%	15.8%±5.6%	27.3%±6.8%	38.3%±7.1%

eps: 2048

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
protoss_5_vs_5	86.7%	10.2%±5.1%	28.1%±4.9%	28.1%±10.4%	39.1%±10.0%	37.3%±4.4%	56.2%±10.6%	65.6%±11.0%	73.4%±5.2%
protoss_10_vs_10	90.3%	2.3%±2.6%	16.4%±3.4%	25.0%±5.9%	43.8%±10.6%	9.4%±1.5%	39.8%±6.4%	45.3%±1.6%	78.1%±5.8%
terran_5_vs_5	81.7%	10.2%±6.0%	28.1%±4.4%	36.7%±7.1%	8.6%±3.4%	7.0%±1.4%	28.9%±5.1%	55.5%±5.1%	56.2%±7.3%
terran_10_vs_10	81.7%	5.5%±2.6%	24.2%±2.6%	43.8%±11.0%	0.0%±0.0%	0.0%±0.0%	0.0%±0.0%	46.9%±8.8%	69.5%±7.1%
zerg_5_vs_5	73.5%	7.0%±4.6%	15.6%±3.8%	26.3%±5.3%	28.1%±3.8%	3.9%±1.5%	33.6%±7.5%	48.1%±8.0%	57.0%±10.2%
zerg_10_vs_10	76.3%	5.5%±2.6%	19.5%±3.4%	10.4%±3.6%	25.8%±8.1%	1.2%±1.3%	46.9%±7.3%	49.2%±2.6%	53.9%±11.8%

eps: 2048

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
miner_easy_2_vs_2	82.4%	35.9%±3.5%	19.5%±1.2%	21.1%±1.4%	43.8%±2.2%	26.6%±1.6%	36.7%±7.1%	50.0%±4.9%	64.8%±6.0%
miner_medium_2_vs_2	74.9%	19.5%±7.1%	9.8%±4.4%	17.2%±3.5%	30.5%±4.6%	21.1%±3.4%	28.5%±2.2%	41.4%±5.1%	43.8%±11.5%
miner_hard_2_vs_2	69.8%	10.2%±3.4%	8.6%±2.0%	10.2%±2.6%	20.7%±2.6%	14.1%±6.8%	18.8%±3.1%	39.8%±8.1%	45.3%±4.7%

eps: 4096

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)	MIFQ (Det-Sigmoid)	MIFQ (Soft-Sigmoid)	MIFQ (Det-Tanh)	MIFQ (Soft-Tanh)
protoss_5_vs_5	86.7%	19.5%±5.6%	33.6%±8.9%	39.8%±4.1%	47.7%±2.5%	42.6%±3.9%	42.6%±4.6%	64.8%±4.1%	72.7%±10.0%	49.2%±4.6%	53.1%±6.6%	60.9%±10.0%	62.5%±3.1%
protoss_10_vs_10	90.3%	5.5%±1.4%	16.6%±0.3%	38.3%±8.9%	36.8%±2.8%	19.8%±2.9%	28.3%±2.3%	61.7%±5.8%	77.3%±5.6%	59.2%±8.2%	45.3%±9.2%	36.9%±7.2%	64.4%±1.1%
terran_5_vs_5	81.7%	18.0%±2.6%	19.7%±5.3%	32.0%±6.0%	24.6%±4.2%	10.9%±4.5%	10.9%±4.5%	55.9%±0.4%	71.9%±3.8%	48.4%±2.9%	35.2%±10.5%	45.1%±2.7%	60.6%±6.7%
terran_10_vs_10	81.7%	6.2%±2.2%	14.1%±1.7%	35.9%±4.7%	0.5%±0.5%	2.3%±1.4%	1.0%±0.7%	53.8%±6.9%	72.7%±3.4%	26.0%±2.2%	46.1%±9.2%	28.6%±2.6%	53.8%±6.7%
zerg_5_vs_5	73.5%	10.9%±3.5%	18.6%±1.4%	33.6%±7.5%	14.8%±3.1%	5.3%±1.2%	18.8%±0.6%	46.7%±5.2%	60.9%±14.7%	39.8%±5.1%	21.1%±5.1%	27.9%±2.2%	43.8%±5.7%
zerg_10_vs_10	76.3%	9.4%±3.8%	16.4%±0.6%	17.2%±3.5%	27.1%±4.8%	1.2%±1.2%	31.2%±3.6%	51.0%±2.2%	59.4%±3.1%	12.5%±4.9%	29.7%±2.7%	15.2%±3.9%	41.3%±2.2%

eps: 4096

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)	MIFQ (Det-Sigmoid)	MIFQ (Soft-Sigmoid)	MIFQ (Det-Tanh)	MIFQ (Soft-Tanh)
miner_easy_2_vs_2	82.4%	31.2%±5.8%	21.9%±4.4%	18.8%±3.8%	36.7%±2.9%	35.2%±2.6%	34.6%±3.3%	52.7%±0.7%	60.2%±5.6%	32.0%±4.6%	53.1%±7.7%	56.4%±7.2%	43.1%±4.5%
miner_medium_2_vs_2	74.9%	28.1%±3.8%	9.8%±1.6%	14.8%±3.4%	28.3%±1.9%	21.1%±2.6%	26.0%±1.5%	43.8%±2.2%	49.2%±5.1%	24.8%±2.2%	40.6%±5.8%	45.7%±10.1%	38.7%±12.6%
miner_hard_2_vs_2	69.8%	12.5%±2.2%	6.6%±0.7%	11.7%±2.6%	19.9%±2.0%	17.2%±6.4%	20.9%±2.4%	39.1%±5.2%	39.8%±4.6%	21.7%±2.0%	25.8%±3.4%	34.8%±3.9%	32.5%±10.8%

```
In [ ]: for eps in all_eps:
        show_table("simple", eps=eps, percentage=False, std=True)
```

eps: 1

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
simple_reference	-17.2	-20.5±0.0	-26.5±0.1	-29.1±0.1	-33.5±0.9	-45.4±2.4	-45.8±0.3	-42.8±0.3	-23.1±0.0
simple_spread	-10.7	-21.7±0.0	-30.3±0.2	-41.6±0.4	-44.2±0.1	-28.9±0.6	-31.8±1.5	-30.4±0.1	-30.8±0.4
simple_speaker_listener	-19.7	-28.5±0.0	-53.8±0.4	-37.7±0.2	-138.9±0.4	-127.4±0.1	-115.6±1.6	-50.5±0.3	-27.6±0.0

eps: 2

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
simple_reference	-17.5	-24.1±0.2	-28.4±0.1	-39.0±1.2	-49.7±0.1	-48.6±0.1	-43.4±0.1	-44.6±0.6	-22.4±0.1
simple_spread	-11.4	-24.3±0.0	-35.3±0.2	-48.1±1.9	-54.8±0.2	-33.6±2.2	-25.2±0.1	-27.5±0.6	-28.4±0.4
simple_speaker_listener	-19.2	-27.6±0.0	-43.7±0.4	-56.4±4.6	-113.9±3.7	-121.5±6.9	-107.6±0.2	-44.2±0.0	-27.3±0.0

eps: 4

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)	
simple	simple_reference	-17.5	-22.5±0.1	-26.2±0.4	-26.8±0.7	-26.7±0.0	-48.2±0.3	-50.3±0.2	-45.2±0.2	-20.0±0.0
	simple_spread	-11.4	-23.8±0.0	-28.5±4.8	-34.2±1.1	-46.9±0.2	-28.0±0.2	-43.5±0.2	-25.9±0.2	-23.1±0.4
	simple_speaker_listener	-19.2	-29.6±0.0	-67.3±1.3	-40.0±1.9	-113.1±0.6	-134.0±0.0	-92.5±1.2	-125.4±0.2	-29.9±0.0

eps: 8

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
simple_reference	-17.5	-19.0±0.1	-25.6±0.1	-35.9±1.6	-43.1±0.6	-46.5±1.0	-42.8±1.1	-33.1±0.3	-20.2±0.1
simple_spread	-11.4	-21.7±0.0	-27.9±0.1	-29.3±0.2	-47.0±1.2	-33.2±1.9	-28.4±0.7	-35.8±0.1	-53.1±0.9
simple_speaker_listener	-19.2	-32.9±0.0	-35.9±0.1	-38.6±0.3	-111.1±0.2	-134.0±0.0	-99.4±0.5	-59.9±0.2	-28.9±0.1

eps: 16

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
simple_reference	-17.5	-18.2±0.1	-20.8±0.2	-22.7±0.2	-32.0±0.5	-47.8±2.1	-45.4±0.6	-38.9±0.2	-18.8±0.1
simple_spread	-11.4	-21.3±0.0	-24.2±0.2	-38.7±1.1	-45.1±0.2	-25.3±0.5	-41.2±1.0	-31.0±0.3	-37.7±0.6
simple_speaker_listener	-19.2	-31.9±0.0	-56.5±0.1	-35.2±0.5	-129.4±0.4	-134.0±0.0	-134.0±0.0	-40.4±0.5	-27.9±0.0

eps: 32

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
simple reference	-17.5	-18.3±0.0	-18.7±0.2	-21.7±0.2	-46.0±0.2	-30.4±1.5	-30.7±1.3	-32.2±0.1	-18.6±0.1
simple spread	-11.4	-20.6±0.0	-27.1±0.1	-34.7±0.5	-34.9±0.3	-32.1±0.7	-33.8±0.4	-24.8±0.2	-54.9±0.4
simple speaker listener	-19.2	-29.0±0.0	-32.8±0.2	-35.9±0.1	-104.4±0.2	-132.5±2.1	-134.8±0.4	-36.8±1.0	-28.1±0.0

eps: 64

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)
simple_reference	-17.5	-18.1±0.0	-18.3±0.2	-18.9±0.2	-36.4±0.7	-31.3±1.0	-42.6±0.2	-24.6±0.1	-20.4±0.0
simple_spread	-11.4	-20.3±0.0	-23.0±0.1	-23.8±0.2	-29.9±0.2	-29.4±0.0	-32.7±0.1	-23.5±0.2	-23.7±0.2
simple_speaker_listener	-19.2	-28.4±0.0	-30.3±0.3	-39.8±1.1	-69.2±1.7	-124.6±0.3	-117.7±0.1	-33.4±0.0	-26.7±0.0

eps: 128

	Expert	BC	IIQ	IQVDN	MASQIL	MAAIRL	MAGAIL	MIFQ (Det)	MIFQ (Soft)	MIFQ (Det-Sigmoid)	MIFQ (Soft-Sigmoid)	MIFQ (Det-Tanh)	MIFQ (Soft-Tanh)	
	simple_reference	-17.2	-18.3±0.0	-18.6±0.2	-19.0±0.1	-36.6±1.5	-40.7±3.2	-40.0±0.2	-23.4±0.1	-20.5±0.0	-40.4±1.2	-23.0±0.1	-22.1±0.1	-21.7±0.1
	simple_spread	-10.7	-20.5±0.0	-23.6±0.2	-21.6±0.2	-23.0±0.4	-26.5±0.3	-24.3±0.1	-23.2±0.1	-24.2±0.1	-46.4±0.1	-24.7±0.7	-27.8±1.1	-26.6±1.5
	simple_speaker_listener	-19.7	-27.5±0.0	-29.1±0.1	-29.5±0.5	-104.3±5.0	-125.5±0.7	-78.7±4.9	-30.8±0.1	-26.3±0.0	-133.4±0.1	-28.9±0.0	-34.7±1.0	-28.2±0.0

```
In [ ]: def create_baseline_fig(data, showlegend_dict, ts_factor=0.85, extra_activation=False):
    if extra_activation:
        data = {k: v for k, v in data.items() if "MIFQ" in k or "TANH" in k or "SIGMOID" in k}
    else:
        data = {k: v for k, v in data.items() if "TANH" not in k and "SIGMOID" not in k}

    fig = go.Figure()
    is_show_expert = False
    for algo, values in data.items():
        showlegend_dict["Expert"] = showlegend_dict.get("Expert", 0) + 1
        showlegend = showlegend_dict["Expert"] <= 1
        fig.add_trace(go.Scatter(
            x=values["steps"], y=values["winrates"],
            mode='lines', showlegend=False, line_color=COLOR_MAPS[algo], opacity=0.15, line_width=1
        ))
    if not is_show_expert:
        fig.add_trace(go.Scatter(
            x=[0, max(values["steps"])], y=[values["expert_winrate"], values["expert_winrate"]],
            mode='lines', showlegend=showlegend, name="Expert", line_color="#595959", line_dash="dash", line_width=1
        ))
    is_show_expert = True
    algos = sorted(data.keys(), key=ALGOS.index)
    for algo in algos:
        showlegend_dict[algo] = showlegend_dict.get(algo, 0) + 1
        showlegend = showlegend_dict[algo] <= 1
        name = ALGO_NAMES[algo.upper()]
        if extra_activation:
            if name == "MIFQ (Det)":
                name = "MIFQ (Det-ELU)"
            elif name == "MIFQ (Soft)":
                name = "MIFQ (Soft-ELU)"
        values = data[algo]
        smooth_winrates = smooth(values["winrates"], ts_factor)
        fig.add_trace(go.Scatter(
            x=values["steps"], y=smooth_winrates,
            mode='lines', showlegend=showlegend, name=name, line_color=COLOR_MAPS[algo], line_width=2 if "ours" in name else 1.6
        ))
    fig.update_layout(template='simple_white', margin=dict(l=0, r=0, t=0, b=0, pad=0, autoexpand=True))
    fig.update_layout(height=120, width=180)
    fig.update_xaxes(minor=dict(ticklen=2))
    return fig

plotly_figs = {}
showlegend_dict = {}
for eps in reversed(all_eps):
    for env_name, values in data[f"eps{eps}"].items():
        if env_name.startswith("miner"):
            name = "miner"
        elif env_name.startswith("simple"):
            name = "simple"
        else:
            name = "smac"
        if name not in showlegend_dict:
            showlegend_dict[name] = {}
        if eps not in plotly_figs:
            plotly_figs[eps] = {}
        plotly_figs[eps][env_name] = create_baseline_fig(values, showlegend_dict[name])

In [ ]: def show_smac(eps, prefix="", save_fig=True):
    env_names = [env_name for env_name in plotly_figs[eps] if any(x in env_name for x in ["protoss", "terran", "zerg"])]
    if len(env_names) == 0:
        return
    print(f"Number of expert episodes: {eps}")
    env_names.sort(key=lambda x: x.replace("5_vs_5", "05_vs_05"))
    os.makedirs("graphs", exist_ok=True)
    fig = make_subplots(rows=1, cols=6, horizontal_spacing=0.06, vertical_spacing=0.2)
    for i, env_name in enumerate(env_names):
        plotly_fig = plotly_figs[eps][env_name]
        plotly_fig.update_xaxes(range=[2e4, 1e6], dtick=5e5, minor=dict(ticklen=3, nticks=3))
```

```

plotly_fig.update_yaxes(range=[-1e-2, 0.95], dtick=0.4, minor=dict(ticklen=3, nticks=2))
plotly_fig.update_layout(showlegend=False)
if save_fig or True:
    plotly_fig.write_image(f"graphs/{env_name}_{eps}{prefix}.pdf")
fig.add_traces(plotly_fig.data, rows=i//6+1, cols=i%6+1)
fig['layout']['f'xaxis[i+1]'].update(title=env_name)
fig['layout']['f'yaxis[i+1]'].update(showticklabels=i%6==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*7)
fig.update_xaxes(range=[2e4, 1e6], dtick=5e5, minor=dict(ticklen=3, nticks=3))
fig.update_yaxes(range=[-1e-2, 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")
return fig

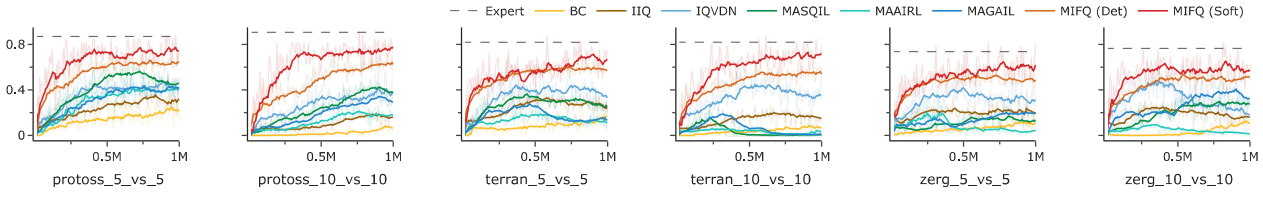
```

```

In [ ]: fig = show_smac(4096)
fig.write_image(f"graphs/legend_smac.pdf")

```

Number of expert episodes: 4096

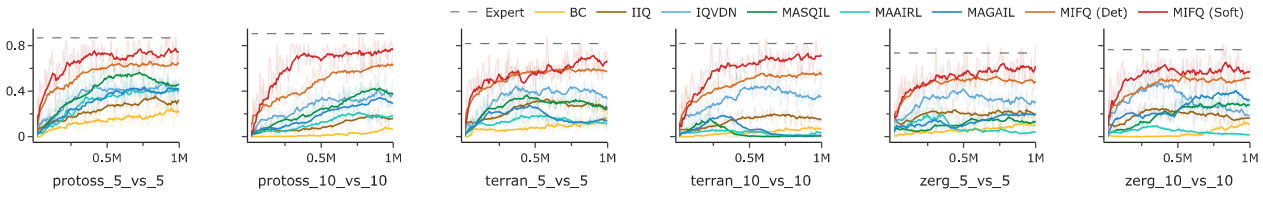


```

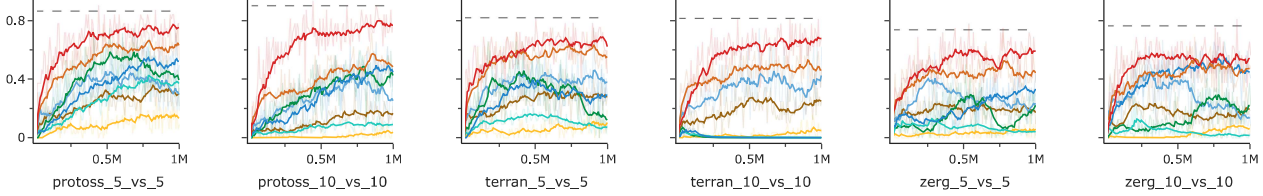
In [ ]: for eps in reversed(all_eps):
fig = show_smac(eps, save_fig=False)

```

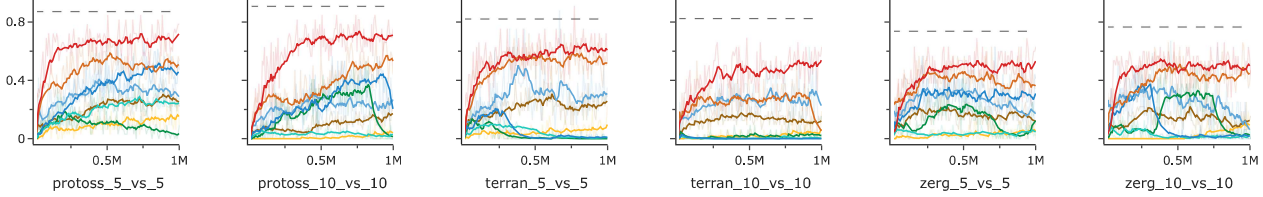
Number of expert episodes: 4096



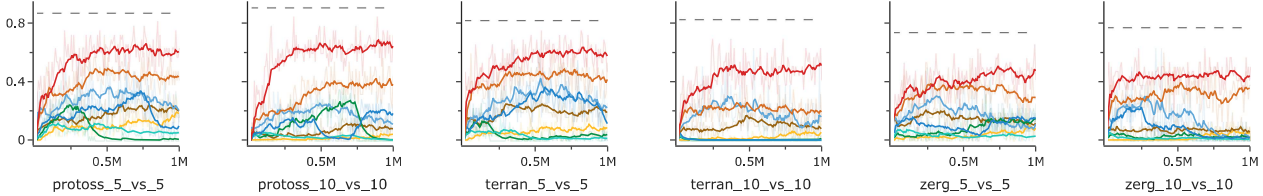
Number of expert episodes: 2048



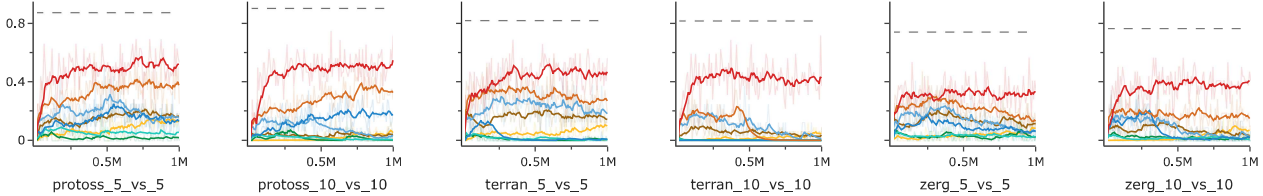
Number of expert episodes: 1024



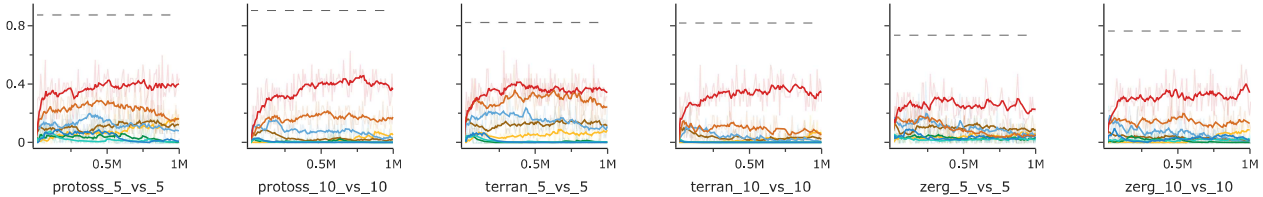
Number of expert episodes: 512



Number of expert episodes: 256



Number of expert episodes: 128



```

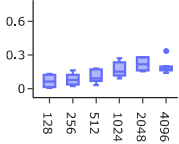
In [ ]: data_df = {}
for eps in all_eps:
    for env_name, item in data[f"eps({eps})"].items():
        if not any(x in env_name for x in ["protoss", "terran", "zerg"]):
            continue
        for algo, values in item.items():
            if algo not in data_df:
                data_df[algo] = {}
            if eps not in data_df[algo]:
                data_df[algo][eps] = {}
            data_df[algo][eps][env_name] = values["winrate"]
for algo in ALGOS:
    if "TANH" in algo or "SIGMOID" in algo or "BC" in algo or algo not in data_df:
        continue
    print("SMAC - algo:", ALGO_NAMES[algo])
    df = pd.DataFrame.from_dict(data_df[algo])
    fig = df.plot.box()

```

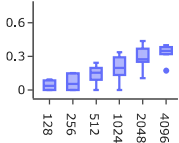


```
fig.update_layout(template='simple_white', margin=dict(l=4, r=4, t=4, b=4, pad=4, autoexpand=True))
fig.update_layout(height=150, width=180)
fig.update_layout(xaxis_title=None, yaxis_title=None)
fig.update_yaxes(range=[-4e-2, 0.79], dtick=0.3, minor=dict(ticklen=3, nticks=2))
fig.write_image(f"graphs/smac_{algo}_box.pdf")
fig.show("svg")
```

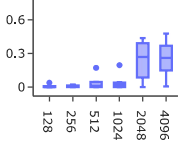
SMAC - algo: IIQ



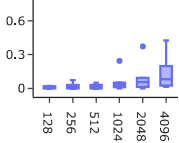
SMAC - algo: IQVDN



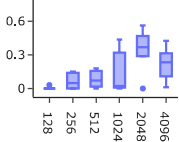
SMAC - algo: MASQIL



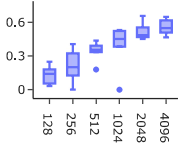
SMAC - algo: MAAIRL



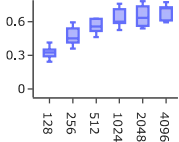
SMAC - algo: MAGAIL



SMAC - algo: MIFQ (Det)



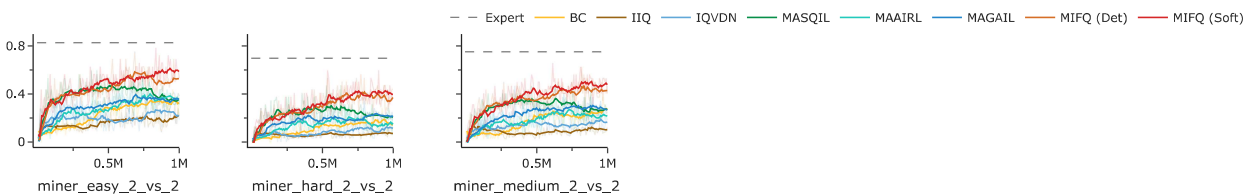
SMAC - algo: MIFQ (Soft)



```
In [ ]: def show_miner(eps, prefix="", save_fig=True):
env_names = [env_name for env_name in plotly_figs[eps] if env_name.startswith("miner")]
if len(env_names) == 0:
return
print(f"Number of expert episodes: {eps}")
os.makedirs("graphs", exist_ok=True)
fig = make_subplots(rows=1, cols=6, horizontal_spacing=0.06, vertical_spacing=0.2)
for i, env_name in enumerate(env_names):
plotly_fig = plotly_figs[eps][env_name]
plotly_fig.update_xaxes(range=[1e2, 1e6], dtick=5e5, minor=dict(ticklen=3, nticks=3))
plotly_fig.update_yaxes(range=[-1e-2, 0.9], dtick=0.4, minor=dict(ticklen=3, nticks=2))
plotly_fig.update_layout(showlegend=False)
if save_fig or True:
plotly_fig.write_image(f"graphs/{env_name}_{eps}{prefix}.pdf")
fig.add_traces(plotly_fig.data, rows=1//6+1, cols=1%6+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*7)
fig.update_xaxes(range=[1e2, 1e6], dtick=5e5, minor=dict(ticklen=3, nticks=3))
fig.update_yaxes(range=[-1e-2, 0.9], 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")
return fig
```

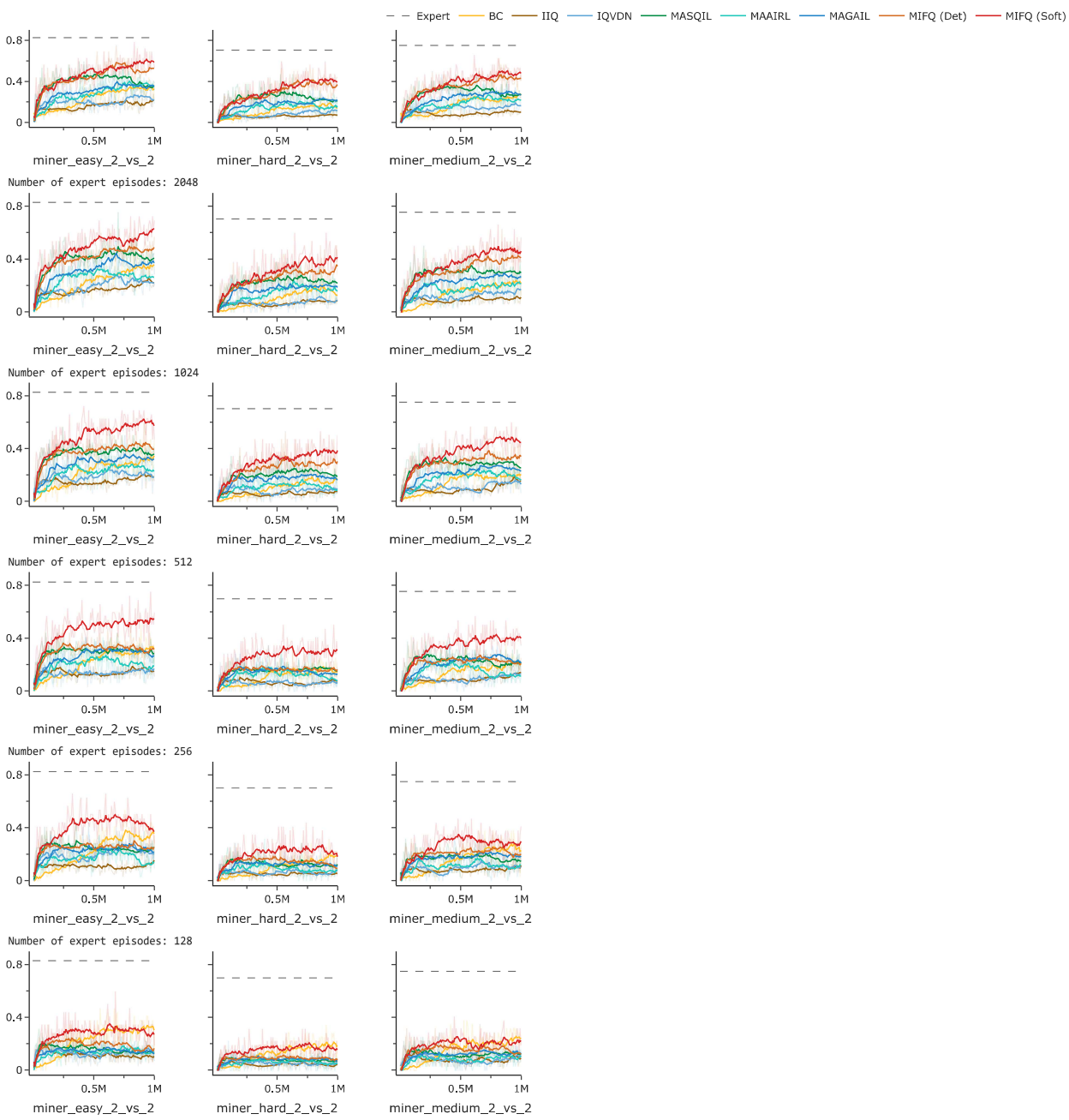
```
In [ ]: fig = show_miner(4096)
fig.write_image(f"graphs/legend_miner.pdf")
```

Number of expert episodes: 4096

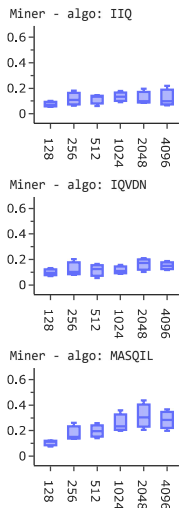


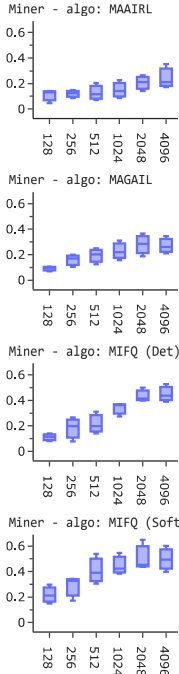
```
In [ ]: for eps in reversed(all_eps):
fig = show_miner(eps, save_fig=False)
```

Number of expert episodes: 4096



```
In [ ]: data_df = {}
for eps in all_eps:
    for env_name, item in data[f"eps{eps}"].items():
        if not env_name.startswith("miner"):
            continue
        for algo, values in item.items():
            if algo not in data_df:
                data_df[algo] = {}
            if eps not in data_df[algo]:
                data_df[algo][eps] = {}
            data_df[algo][eps][env_name] = values["winrate"]
for algo in ALGOS:
    if "TANH" in algo or "SIGMOID" in algo or "BC" in algo or algo not in data_df:
        continue
    print("Miner - algo:", ALGO_NAMES[algo])
    df = pd.DataFrame.from_dict(data_df[algo])
    fig = df.plot.box()
    fig.update_layout(template='simple_white', margin=dict(l=4, r=4, t=4, b=4, pad=4, autoexpand=True))
    fig.update_layout(height=150, width=180)
    fig.update_layout(xaxis_title=None, yaxis_title=None)
    fig.update_yaxes(range=[-4e-2, 0.69], dtick=0.2, minor=dict(ticklen=3, nticks=2))
    fig.write_image(f"graphs/miner_{algo}_box.pdf")
    fig.show("svg")
```

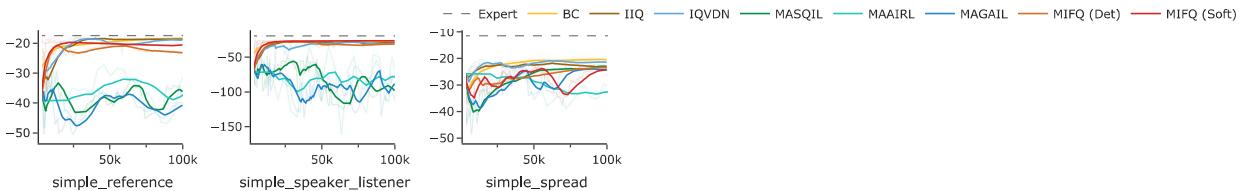




```
In [ ]: def show_mpe(eps, prefix="", save_fig=True):
env_names = [env_name for env_name in plotly_figs[eps] if env_name.startswith("simple")]
if len(env_names) == 0:
return
print(f"Number of expert episodes: {eps}")
os.makedirs("graphs", exist_ok=True)
fig = make_subplots(rows=1, cols=6, horizontal_spacing=0.06, vertical_spacing=0.2)
for i, env_name in enumerate(env_names):
plotly_fig = plotly_figs[eps][env_name]
plotly_fig.update_xaxes(range=[4e3, 1e5], dtick=5e4, minor=dict(ticklen=3, nticks=3))
# plotly_fig.update_yaxes(range=[-1e-2, 0.59], dtick=0.2, minor=dict(ticklen=3, nticks=2))
plotly_fig.update_layout(showlegend=False)
if save_fig or True:
plotly_fig.write_image(f"graphs/{env_name}_{eps}(prefix).pdf")
fig.add_traces(plotly_fig.data, rows=1//6+1, cols=1*6+1)
fig["layout"]["xaxis[i+1]"].update(title=env_name)
fig.update_layout(template="simple_white", margin=dict(l=4, r=4, b=4, pad=4, autoexpand=True))
fig.update_layout(height=200, width=180*7)
fig.update_xaxes(range=[4e3, 1e5], dtick=5e4, minor=dict(ticklen=3, nticks=3))
fig.update_layout(legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1))
fig.update_layout(autosize=False)
fig.show("svg")
return fig
```

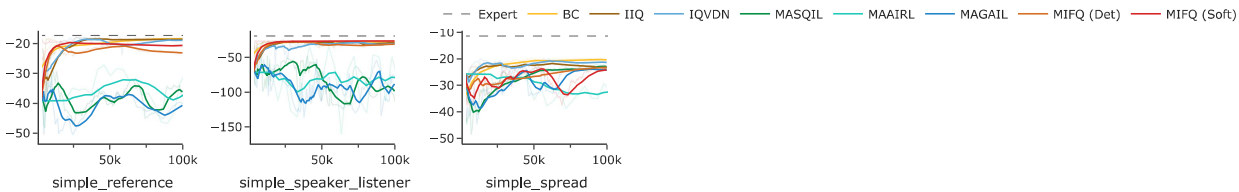
```
In [ ]: fig = show_mpe(128)
fig.write_image(f"graphs/legend_mpe.pdf")
```

Number of expert episodes: 128

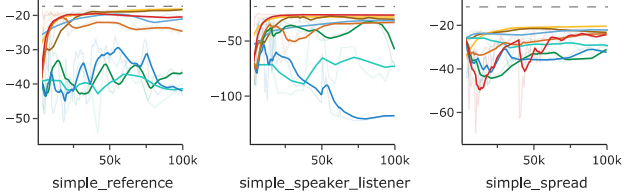


```
In [ ]: for eps in reversed(all_eps):
fig = show_mpe(eps, save_fig=False)
```

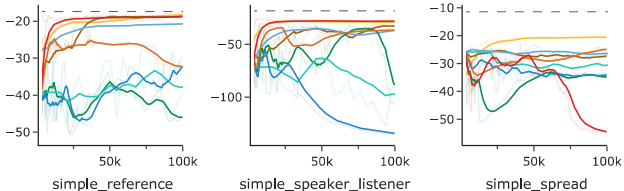
Number of expert episodes: 128



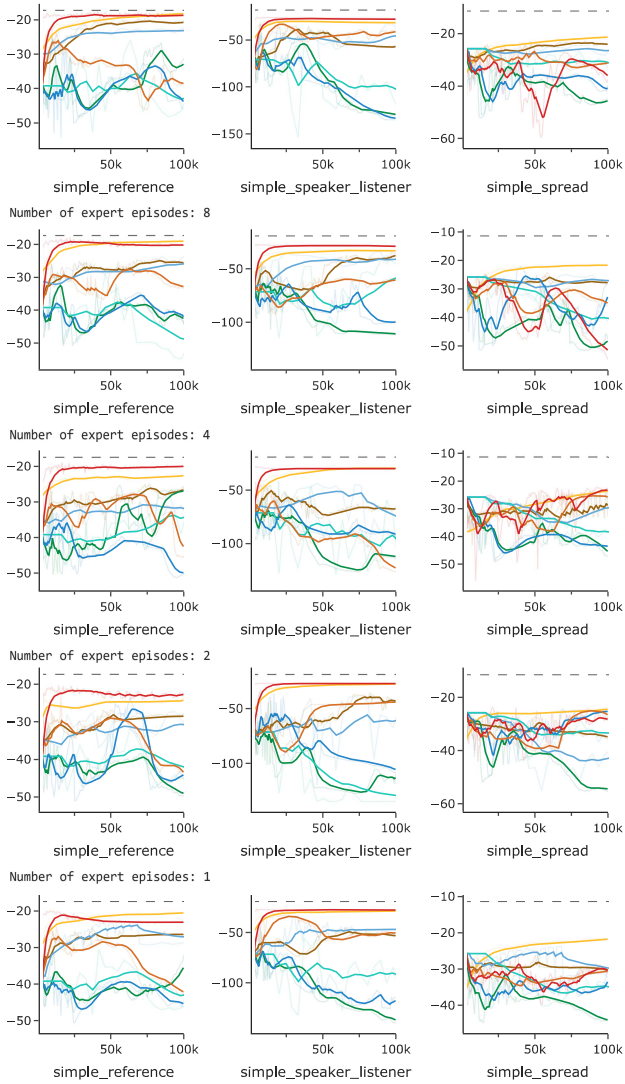
Number of expert episodes: 64



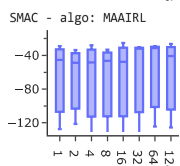
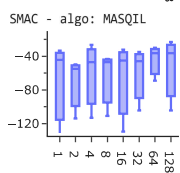
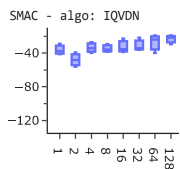
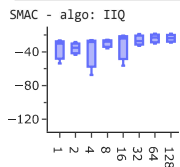
Number of expert episodes: 32



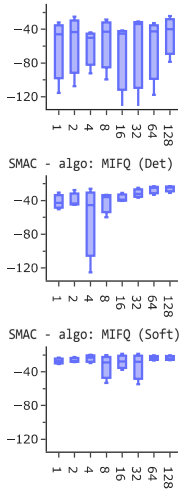
Number of expert episodes: 16



```
In [ ]: data_df = {}
for eps in all_eps:
    for env_name, item in data[f"eps={eps}"].items():
        if not env_name.startswith("simple"):
            continue
        for algo, values in item.items():
            if algo not in data_df:
                data_df[algo] = {}
            if eps not in data_df[algo]:
                data_df[algo][eps] = {}
            data_df[algo][eps][env_name] = values["winrate"]
for algo in ALGOS:
    if "TANH" in algo or "SIGMOID" in algo or "BC" in algo or algo not in data_df:
        continue
    print("SMAC - algo:", ALGO_NAMES[algo])
    df = pd.DataFrame.from_dict(data_df[algo])
    fig = df.plot.box()
    fig.update_layout(template='simple_white', margin=dict(l=4, r=4, t=4, b=4, pad=4, autoexpand=True))
    fig.update_layout(height=150, width=180)
    fig.update_layout(xaxis_title=None, yaxis_title=None)
    fig.update_yaxes(range=[-130, -10], dtick=40, minor=dict(ticklen=3, nticks=2))
    fig.write_image(f"graphs/mpe_{algo}_box.pdf")
    fig.show("svg")
```



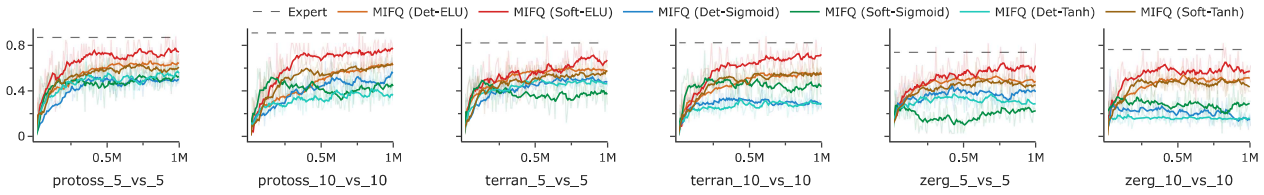
SMAC - algo: MAGAIL



```
In [ ]: plotly_figs = {}
showlegend_dict = {}
for eps in reversed(all_eps):
    for env_name, values in data[f"eps{eps}"].items():
        name = "smac"
        if env_name.startswith("miner"):
            name = "miner"
        elif env_name.startswith("simple"):
            name = "simple"
        if name not in showlegend_dict:
            showlegend_dict[name] = {}
        if eps not in plotly_figs:
            plotly_figs[eps] = {}
        plotly_figs[eps][env_name] = create_baseline_fig(values, showlegend_dict[name], extra_activation=True)
```

```
In [ ]: fig = show_smac(4096, "_extend")
fig.write_image(f"graphs/legend_smac_extend.pdf")
```

Number of expert episodes: 4096



```
In [ ]: fig = show_miner(4096, "_extend")
fig.write_image(f"graphs/legend_miner_extend.pdf")
```

Number of expert episodes: 4096

