Plot and TensorBoard

Ramin Zarebidoky (LiterallyTheOne)

27 Aug 2025

Plot and TensorBoard

Introduction

In the previous tutorials, we were just printing our training and evaluation results. When our training epochs become larger or when we want to compare two methods with each other, looking at the numbers becomes devastating. One of the best ways to do that is to plot them. In this tutorial, we are going to first plot the results using `matplotlib`, then we will be using `TensorBoard` to achieve a better result.

Plot using matplotlib

To plot our results using `matplotlib`, the first thing that we should do is to make a list of our previous results in our training loop, like below:

```
train_losses = []
train_accuracies = []

val_losses = []
val_accuracies = []

for epoch in range(20):
    print("-" * 20)
    print(f"epoch: {epoch}")

    train_loss, train_accuracy = train_step(train_loader, model,
↪   optimizer, loss_fn, device)
    train_losses.append(train_loss)
    train_accuracies.append(train_accuracy)

    val_loss, val_accuracy = val_step(val_loader, model, loss_fn,
↪   device)
    val_losses.append(val_loss)
    val_accuracies.append(val_accuracy)
```

```
    print(f"train: ")
    print(f"\tloss: {train_loss:.4f}")
    print(f"\taccuracy: {train_accuracy:.4f}")

    print(f"validation: ")
    print(f"\tloss: {val_loss:.4f}")
    print(f"\taccuracy: {val_accuracy:.4f}")
```

In the code above, I have created 4 lists (`train_losses`, `train_accuracies`, `val_losses`, `val_accuracies`). Each list is for a different result. As you can see, I have increased the epoch range to `20` as well. Now, let's plot our results.

```
from matplotlib import pyplot as plt

# -------------------[ Plot our results ]--------------------
plt.figure()
plt.title("loss")
plt.plot(train_losses, label="train")
plt.plot(val_losses, label="val")
plt.legend()

plt.figure()
plt.title("accuracy")
plt.plot(train_accuracies, label="train")
plt.plot(val_accuracies, label="val")
plt.legend()

plt.show()
```

In the code above, I plot losses and accuracies in different figures. I have put all the changed parts in train_plot.py. So, the output would be something like below:

As you can see, analyzing the plots is so much easier than examining the numbers.

TensorBoard

`TensorBoard` is one of the most used and greatest tools to keep track of our training. It has so many features, but we are going to focus on only plotting. To do so, we should have a `TensorBoard writer`. So, let's make one.

```
from torch.utils.tensorboard import SummaryWriter

# -------------------[ Setup TensorBoard ]--------------------
writer = SummaryWriter()
```

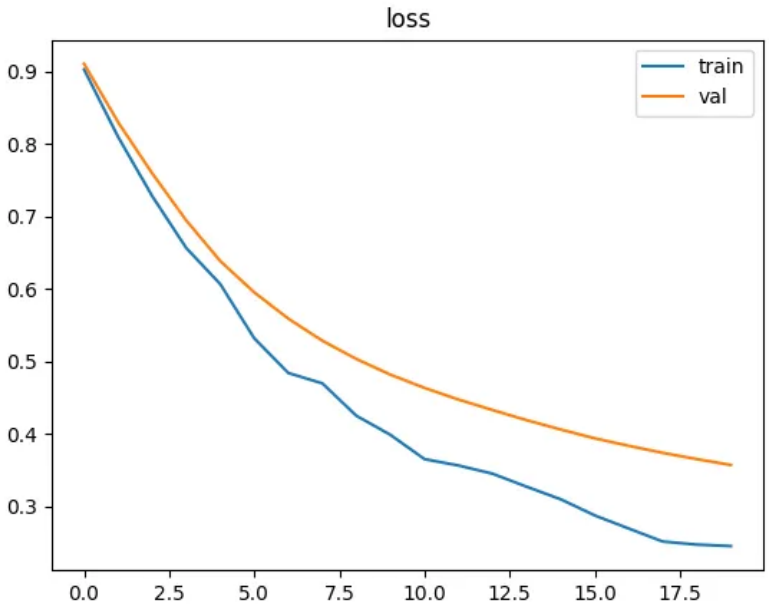

Figure 1: Loss

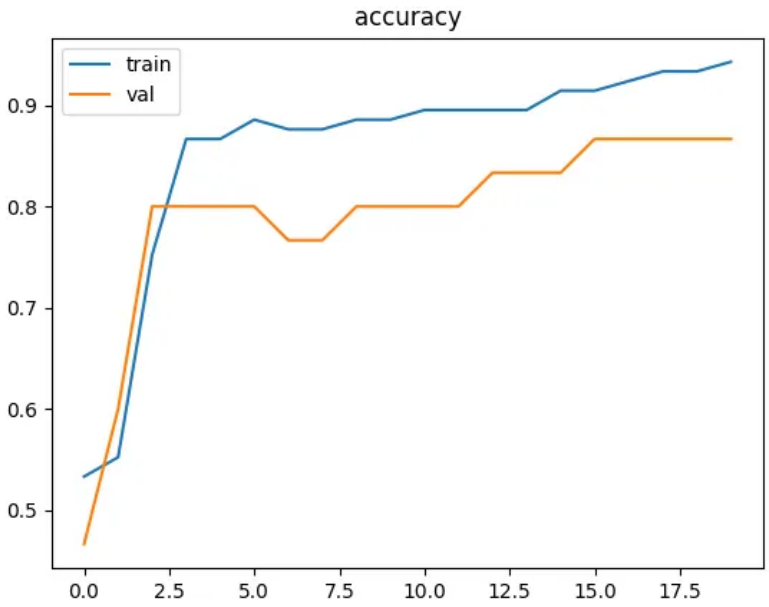

Figure 2: Accuracy

In the code above, I have imported the `SummaryWriter` from `torch.utils.tensorboard`. Then, I have created an instance of `SummaryWriter` and called it `writer`. By default, `SummaryWriter` creates a log directory with the name of `runs` and stores the data into that directory. Now, let's write our training and evaluation results with `writer`.

```
# --------------------[ Train and evaluate the model
 ↪  ]--------------------
for epoch in range(20):
    print("-" * 20)
    print(f"epoch: {epoch}")

    train_loss, train_accuracy = train_step(train_loader, model,
 ↪  optimizer, loss_fn, device)
    writer.add_scalar("loss/train", train_loss, epoch)
    writer.add_scalar("accuracy/train", train_accuracy, epoch)

    val_loss, val_accuracy = val_step(val_loader, model, loss_fn,
 ↪  device)
    writer.add_scalar("loss/val", val_loss, epoch)
    writer.add_scalar("accuracy/val", val_accuracy, epoch)

    print(f"train: ")
    print(f"\tloss: {train_loss:.4f}")
    print(f"\taccuracy: {train_accuracy:.4f}")

    print(f"validation: ")
    print(f"\tloss: {val_loss:.4f}")
    print(f"\taccuracy: {val_accuracy:.4f}")
```

As you can see, in our `training loop`, I used the `add_scalar` function of the `writer` to write the results. For each result, I have chosen different names.

- `train_loss` -> `loss/train`
- `train_accuary` -> `loss/accuracy`
- `val_loss` -> `loss/val`
- `val_accuary` -> `loss/accuracy`

I have applied the changes to train_tensorboard.py. Now, let's write our training script multiple times. For example, I have run it `3` times. After that, let's run our `TensorBoard` to see the results. To do so, we can run the command below:

```
tensorboard --log_dir runs
```

Or if you want to see the results on a notebook, you can use the code below:

```
%load_ext tensorboard
%tensorboard --logdir runs
```

The output would be something like below:

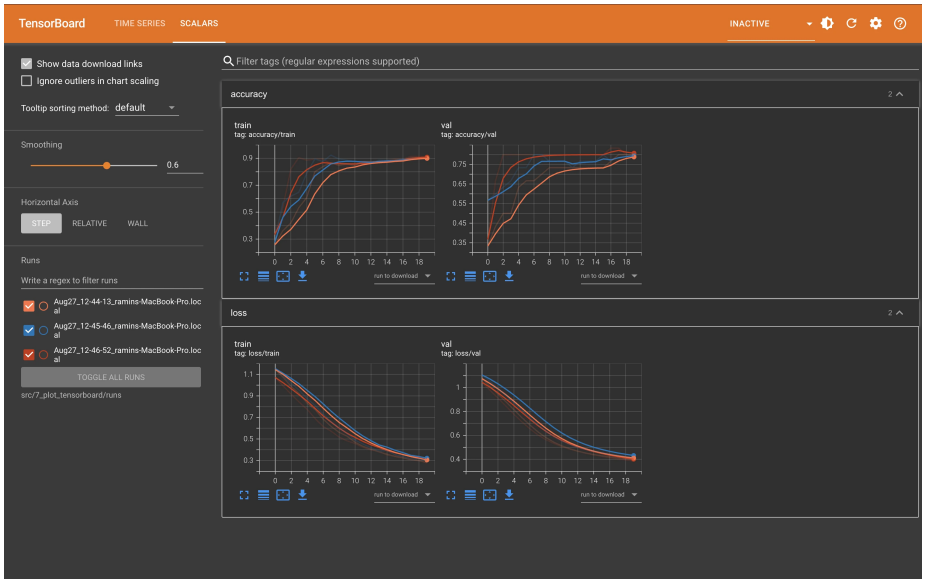

Figure 3: tensorboard result

As you can see, my `3` runs are being displayed separately. Right now, I can easily analyze my training. Also, because I write the results into files, I can see the results during the training process, which is extremely helpful

Conclusion

In this tutorial, we have learned how to plot our training and evaluation results. First, we plotted our results using `matplotlib`. Then, we learned how to use `TensorBoard` for better analysis.