Introduction to Deep Learning

Tutorial 8

Gabriel Deza
Department of Industrial Engineering
Tel Aviv University

Deceember 23 & 25, 2025

Topics for Today

Exam questions & discussion

- Cover questions of all *kinds* on topics we have learned so far
- Questions

Course roadmap

Weeks 1–8 (so far)

- **Models**: linear regression, classification, MLP, CNNs, pre-trained
- **Losses**: MSE, CE, BCE
- **Activations**: ReLU, Tanh, sigmoid, softmax
- **Backpropagation**
- **Optimizers**: (S)GD, Momentum, Adam
- **Regularization**: batch norm, dropout, weight decay
- **Hyperparameter tuning**: Optuna, W&B
- **Vision**: CNNs, receptive fields
- **Transfer learning**: fine-tuning, style transfer

Next $\approx$ 5 weeks

- Sequence models: RNN, LSTM, GRU
- Recommender systems
- Language models
- Attention & Transformers
- Generative deep learning: GANs, autoencoders
- LLMs

PyTorch throughout

Exam Questions

- Let's start with easy questions and increase in difficulty from 1 to 10.

Question 1

Code

```
import torch
from torch import nn

n_input = 10
n_hidden = 7
n_output = 1

net = nn.Sequential(
    nn.Linear(n_input, n_hidden),
    nn.ELU(),
    nn.Linear(n_hidden, n_output),
    nn.Sigmoid())

x = torch.randn(100, 10)
y = net(x)
print(y.shape)
```

Question

The code builds a network and runs a 2D input matrix.

What is the output's shape?

(a) `torch.Size([10, 1])`

(b) `torch.Size([7, 10])`

(c) `torch.Size([1, 100])`

(d) `torch.Size([1, 10])`

(e) None of the above.

Question 2

18. In respect to epochs and batches, mark all that applies:

(a) Epoch and batch mean the same thing.

(b) The epoch iterations happen within the batch iteration.

(c) The batch iterations happen within the epoch iteration.

(d) Epoch is an iteration over the entire data points, while batches are subsets of the data points.

(e) Epoch is an iteration over the entire **training** data points, while batches are iterations over the entire **test** data points.

(f) In a single batch, the weights (learnable parameters) of the network are adjusted X times, where X is the batch size.

(g) In a single epoch, the weights (learnable parameters) of the network are adjusted X times, where X is the number of batches.

(h) In a single epoch, the weights (learnable parameters) of the network are adjusted X times, where X is the batch size.

Question 3 &4

True or False: Using Dropout during test time can help reduce overfitting:

1. True
2. False

In case of overfitting, which of the following may help (check all that apply)?

1. Add regularization.
2. Get more training data
3. Use data augmentation techniques
4. Remove some hidden layers

Question 5

12. Consider a network with five layers. The input is the first layer, followed by 4 convolutional layers, each employing 3×3 filters (stride 1, no pooling). What is the size of the receptive field in the first (input) layer with respect to an activation at the fifth layer?

(a) 3×3

(b) 5×5

(c) 7×7

(d) 9×9

Question 6

Where will more parameters be trained (in the training process)?

(a)

```
for param in vgg16.features.parameters():
    param.requires_grad = False
for param in vgg16.features[-6:].parameters():
    param.requires_grad = True
```

(b)

```
for param in vgg16.features.parameters():
    param.requires_grad = False
for param in vgg16.features[-5:].parameters():
    param.requires_grad = True
```

Question 7

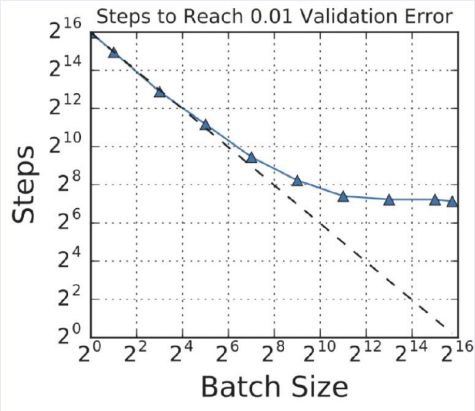
You are a bad prompt engineer . . . ChatGPT wrote you this code — any mistakes?

```
def objective(study):
    learning_rate = trial.suggest_loguniform("learning_rate", 1e-5, 1e-3)
    weight_decay = trial.suggest_float("weight_decay", 1e-6, 1e-4)
    batch_size = trial.suggest_int("batch_size", 16, 64, step=16)

    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    val_loader = DataLoader(val_dataset, batch_size=len(val_dataset), shuffle=True)
    # Load the pre-trained model VGG16
    model = models.vgg16(pretrained=True)
    model = model.to(device)
    for param in model.features.parameters():
        param.requires_grad = False
    for param in model.features[:0].parameters():
        param.requires_grad = True
    # Modify the classifier to fit our problem (10 classes)
    model.classifier[6] = nn.Linear(4096, 10)
    model.classifier[6] = model.classifier[6].to(device)
    criterion = nn.MSELoss()
    optimizer = optim.Adam(model.parameters(), lr=learning_rate, weight_decay=weight_decay)
    best_val_loss = train_model_with_hyperparams(
        model, train_loader, val_loader, optimizer, criterion,
        epochs=20, patience=20, trial=trial)
    return best_val_loss
# Define and optimize the Optuna study
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=20)
```

Question 8

The following plots the number of SGD iterations required to reach a given loss, as a function of the batch size:



(a) For small batch sizes, the number of iterations required to reach the target loss decreases as the batch size increases. Why is that?

(b) For large batch sizes, the number of iterations does not change much as the batch size is increased. Why is that?

Question 9

Recall the logistic activation function σ and the tanh activation function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \qquad \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}.$$

Both activation functions have a sigmoidal shape.

Give the Jacobian matrix $\partial \mathbf{y} / \partial \mathbf{z}$ of the tanh activation function, applied elementwise to all units in a layer. You may give your answer in terms of $\tanh'(z)$, the univariate derivative of tanh.

Q9 (cont.)

One of the difficulties with the logistic activation function is that of *saturated* units. Briefly explain the problem, and whether switching to tanh fixes the problem. (You may refer to your answer from part (a) or sketch the activation functions.)

Any questions?

See you next week for new content :)