BIOS694 Final Project - KD (Alpha = 0.7)

2025-04-17

```
library(torch)
library(torchvision)
library(luz)
library(reshape2)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tibble)
library(caret)
```

```
## Loading required package: lattice
```

```
library(here)
```

```
## here() starts at C:/Users/mince/Desktop/McGill/Courses/BIOS694/Project
```

MNIST dataset

```
torch_manual_seed(42)
dir <- "./dataset/mnist"

train_ds <- mnist_dataset(
  dir,
  download = TRUE,
  transform = transform_to_tensor
)

test_ds <- mnist_dataset(
```

```
  dir,
  train = FALSE,
  transform = transform_to_tensor
)
```

Student model

Define the Student model

```
# No temperature here
StudentNet <- nn_module(
  "StudentNet",

  initialize = function() {
    self$T <- T  # Temperature
    self$conv1 <- nn_conv2d(1, 8, kernel_size = 3, padding = 1)
    self$conv2 <- nn_conv2d(8, 16, kernel_size = 3, padding = 1)

    self$dropout <- nn_dropout2d(0.25)

    self$fc1 <- nn_linear(3136, 32)
    self$fc2 <- nn_linear(32, 10)
  },

  forward = function(x) {
    x %>%                                   # N * 1 * 28 * 28
      self$conv1() %>%                      # N * 8 * 28 * 28
      nnf_relu() %>%
      self$conv2() %>%                      # N * 16 * 28 * 28
      nnf_relu() %>%
      nnf_max_pool2d(kernel_size = 2) %>%   # N * 16 * 14 * 14
      self$dropout() %>%
      torch_flatten(start_dim = 2) %>%      # N * 3136
      self$fc1() %>%                        # N * 32
      nnf_relu() %>%
      self$fc2()                            # N * 10 (logits)
  }
)
```

Knowledge distillation

Load the Teacher model

```
# The teacher was trained with T = 3
fitted_teacher <- luz_load(here("model/mnist-cnn-teacher.pt"))
train_dl_noshuffle <- dataloader(train_ds, batch_size = 128, shuffle = FALSE)
teacher_logits <- predict(fitted_teacher, train_dl_noshuffle)

# Convert to soft targets (teacher probabilities)
teacher_probs <- nnf_softmax(teacher_logits, dim = 2)
```

```
# For testing
test_noshuffle <- dataloader(test_ds, batch_size = 128, shuffle = FALSE)
test_logits <- predict(fitted_teacher, test_noshuffle)

# Convert to soft targets (teacher probabilities)
test_teacher_probs <- nnf_softmax(test_logits, dim = 2)
```

Define the KD loss function

```
kd_loss <- function(student_logits, teacher_probs, true_labels, Temp = 3,
                    alpha = 0.1) {

  # Hard target loss (standard CE loss)
  ce_loss <- nn_cross_entropy_loss()(student_logits, true_labels)

  # Soft target loss (KL divergence between soft predictions)
  student_soft <- nnf_log_softmax(student_logits/Temp, dim = 2)
  teacher_soft <- teacher_probs  # assumed already softmaxed with same Temp

  kl_div <- nnf_kl_div(
    input = student_soft,
    target = teacher_soft,
    reduction = "batchmean"
  )

  # Combine both losses
  total_loss <- alpha * ce_loss + (1 - alpha) * (kl_div * Temp^2)
  return(total_loss)
}
```

Create a custom dataset that includes soft targets

```
# Pull images and labels from the dataset and convert to tensors
train_images <- torch_tensor(train_ds$data, dtype = torch_float())  # shape: N x 28 x 28
train_labels <- torch_tensor(as.numeric(train_ds$targets), dtype = torch_long())  # shape: N
test_images <- torch_tensor(test_ds$data, dtype = torch_float())  # shape: N x 28 x 28
test_labels <- torch_tensor(as.numeric(test_ds$targets), dtype = torch_long())  # shape: N

kd_dataset <- dataset(
  name = "KDDataset",

  initialize = function(images, labels, soft_targets) {
    self$images <- images
    self$labels <- labels
    self$soft_targets <- soft_targets
  },

  .getitem = function(i) {
    list(
      x = self$images[i,..],         # 28 x 28 image
```

```
      y = self$labels[i],            # scalar label
      soft = self$soft_targets[i,..] # vector of length 10
    )
  },

  .length = function() {
    self$images$size()[[1]]  # number of observations
  }
)

kd_ds <- kd_dataset(train_images, train_labels, teacher_probs)
kd_dl <- dataloader(kd_ds, batch_size = 128, shuffle = TRUE)

test_ds_new <- kd_dataset(test_images, test_labels, test_teacher_probs)
test_dl <- dataloader(test_ds_new, batch_size = 128, shuffle = FALSE)
```

```
# Check
match_count <- 0
n <- 1000

for (i in 1:n) {
  true_label <- kd_ds$labels[i] %>% as_array()
  teacher_pred <- torch_argmax(kd_ds$soft_targets[i]) %>% as_array()

  if (true_label == teacher_pred) {
    match_count <- match_count + 1
  }
}

match_count # 988/1000 hard and soft targets match
```

```
## [1] 988
```

KD Attempt 1: Train the Student model with Alpha = 0.7

```
student_kd <- StudentNet()$to(device = "cpu")
optimizer <- optim_adam(student_kd$parameters)

epochs <- 3
Temp <- 3
alpha <- 0.7
```

```
for (epoch in 1:epochs) {
  student_kd$train()
  total_loss <- 0
  correct <- 0
  total <- 0

  coro::loop(for (batch in kd_dl) {
    optimizer$zero_grad()
```

```
    # Inputs and targets
    x <- batch$x$unsqueeze(2)$to(device = "cpu")  # [B, 1, 28, 28]
    y <- batch$y$to(device = "cpu")
    soft <- batch$soft$to(device = "cpu")

    # Forward pass
    logits <- student_kd(x)

    # Knowledge distillation loss
    loss <- kd_loss(logits, soft, y, Temp = Temp, alpha = alpha)

    loss$backward()
    optimizer$step()

    # Hard label accuracy tracking
    pred_classes <- torch_argmax(logits, dim = 2)
    correct <- correct + (pred_classes == y)$sum()$item() # correct predictions
    total <- total + y$size(1)
    total_loss <- total_loss + loss$item()
  })

  acc <- correct/total
  cat(sprintf("Epoch %d | Loss: %.4f | Accuracy: %.2f%%\n",
              epoch, total_loss, acc * 100))
}
```

```
## Epoch 1 | Loss: 460.7033 | Accuracy: 91.87%
## Epoch 2 | Loss: 130.0580 | Accuracy: 97.33%
## Epoch 3 | Loss: 97.7819 | Accuracy: 97.92%
```

Evaluate the model on the test set

```
student_kd$eval()
correct_student <- 0
correct_teacher <- 0
total <- 0

coro::loop(for (batch in test_dl) {
  # Inputs and targets
  x <- batch$x$unsqueeze(2)$to(dtype = torch_float(), device = "cpu")
  y <- batch$y$to(dtype = torch_long(), device = "cpu")
  soft <- batch$soft$to(dtype = torch_float(), device = "cpu")

  # Student predictions
  logits <- student_kd(x)
  student_preds <- torch_argmax(logits, dim = 2)

  # Teacher predictions
  teacher_preds <- torch_argmax(soft, dim = 2)

  # Accuracy comparison
```

```
  correct_student <- correct_student + (student_preds == y)$sum()$item()
  correct_teacher <- correct_teacher + (teacher_preds == y)$sum()$item()
  total <- total + y$size(1)
})

cat(sprintf("Test accuracy of KD student: %.2f%%\n",
            100 * correct_student/total))
```

```
## Test accuracy of KD student: 98.11%
```

```
cat(sprintf("Teacher accuracy on the test set: %.2f%%\n",
            100 * correct_teacher/total))
```

```
## Teacher accuracy on the test set: 98.75%
```

```
torch_save(student_kd, here("model/mnist-student-kd-0.7.pt"))
```