Weekly Summary Template

Brady Miller

Table of contents

Tuesday, April 11 1
Tuesday April 11 1
Thursday, April 13 17

Tuesday, April 11

> **!** TIL
>
> Include a *very brief* summary of what you learnt in this class here.
> Today, I learnt the following concepts in class:
>
> 1. Application of neural networks to breast cancer & titanic datasets
> 2. What are DataLoaders and its strengths
> 3. How to use DataLoaders to determine the value of a handwritten number

Tuesday April 11

```
packages <- c(
    # Old packages
    "ISLR2",
    "dplyr",
    "tidyr",
    "readr",
    "purrr",
    "repr",
    "tidyverse",
    "kableExtra",
```

```
    "IRdisplay",
    # NEW
    "torch",
    "torchvision",
    "luz"
)

# renv::install(packages)
sapply(packages, require, character.only=TRUE)
```

```
Loading required package: ISLR2
```

```
Loading required package: 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
```

```
Loading required package: tidyr
```

```
Loading required package: readr
```

```
Loading required package: purrr
```

```
Loading required package: repr
```

```
Loading required package: tidyverse
```

```
Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
logical.return = TRUE, : there is no package called 'tidyverse'
```

```
Loading required package: kableExtra
```

```
Attaching package: 'kableExtra'

The following object is masked from 'package:dplyr':

    group_rows

Loading required package: IRdisplay

Loading required package: torch

Loading required package: torchvision

Loading required package: luz
```

```
      ISLR2       dplyr       tidyr       readr       purrr        repr
       TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
  tidyverse  kableExtra   IRdisplay       torch torchvision         luz
      FALSE        TRUE        TRUE        TRUE        TRUE        TRUE
```

Application of neural networks to breast cancer & Titanic datasets

```
url <- "https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv"

df <- read_csv(url) %>%
    mutate_if(\(x) is.character(x), as.factor) %>%
    mutate(y = Survived) %>%
    select(-c(Name, Survived)) %>%
    (\(x) {
        names(x) <- tolower(names(x))
        x
    })
```

```
Rows: 887 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): Name, Sex
dbl (6): Survived, Pclass, Age, Siblings/Spouses Aboard, Parents/Children Ab...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsi

# col_names <- c("id", "diagnosis", paste0("feat", 1:30))

# df <- read_csv(
#         url, col_names, col_types = cols()
#     ) %>%
#     select(-id) %>%
#     mutate(y = ifelse(diagnosis == "M", 1, 0)) %>%
#     select(-diagnosis)


# df %>% head
```

```r
k <- 5

test_ind <- sample(
    1:nrow(df),
    floor(nrow(df) / k),
    replace=FALSE
)

df_train <- df[-test_ind, ]
df_test  <- df[test_ind, ]

nrow(df_train) + nrow(df_test) == nrow(df)
```

```
[1] TRUE
```

```r
fit_glm <- glm(
    y ~ .,
    df_train %>% mutate_at("y", factor),
    family = binomial()
)

glm_test <- predict(
    fit_glm,
    df_test,
    output = "response"
)
```

```r
glm_preds <- ifelse(glm_test > 0.5, 1, 0)
table(glm_preds, df_test$y)
```

```
glm_preds  0  1
        0 98 26
        1  9 44
```

```r
NNet <- nn_module(
  initialize = function(p, q1, q2, q3) {
    self$hidden1 <- nn_linear(p, q1)
    self$hidden2 <- nn_linear(q1, q2)
    self$hidden3 <- nn_linear(q2, q3)
    self$output <- nn_linear(q3, 1)
    self$activation <- nn_relu()
    self$sigmoid <- nn_sigmoid()
  },

  forward = function(x) {
    x %>%
      self$hidden1() %>% self$activation() %>%
      self$hidden2() %>% self$activation() %>%
      self$hidden3() %>% self$activation() %>%
      self$output() %>% self$sigmoid()
  }
)
```

```r
# fitting a model without an intercept - when its zero, everything else is zero
M <- model.matrix(y ~ 0 + ., data = df_train)
# model.matrix(y ~ ., data = df_train) [ ,-1]

fit_nn <- NNet %>%
    #
    # Setup the model
    #
    setup(
        loss = nn_bce_loss(),
        optimizer = optim_adam,
        metrics = list(
            luz_metric_accuracy()
        )
    ) %>%
```

```
    #
    # Set the hyperparameters
    #
    set_hparams(p=ncol(M), q1=256, q2=128, q3=64) %>%
    set_opt_hparams(lr=0.005) %>%
    #
    # Fit the model
    #
    fit(
        data = list(
            model.matrix(y ~ 0 + ., data = df_train),
            df_train %>% select(y) %>% as.matrix
        ),
        valid_data = list(
            model.matrix(y ~ 0 + ., data = df_test),
            df_test %>% select(y) %>% as.matrix
        ),
        epochs = 50,
        verbose = TRUE
    )
```

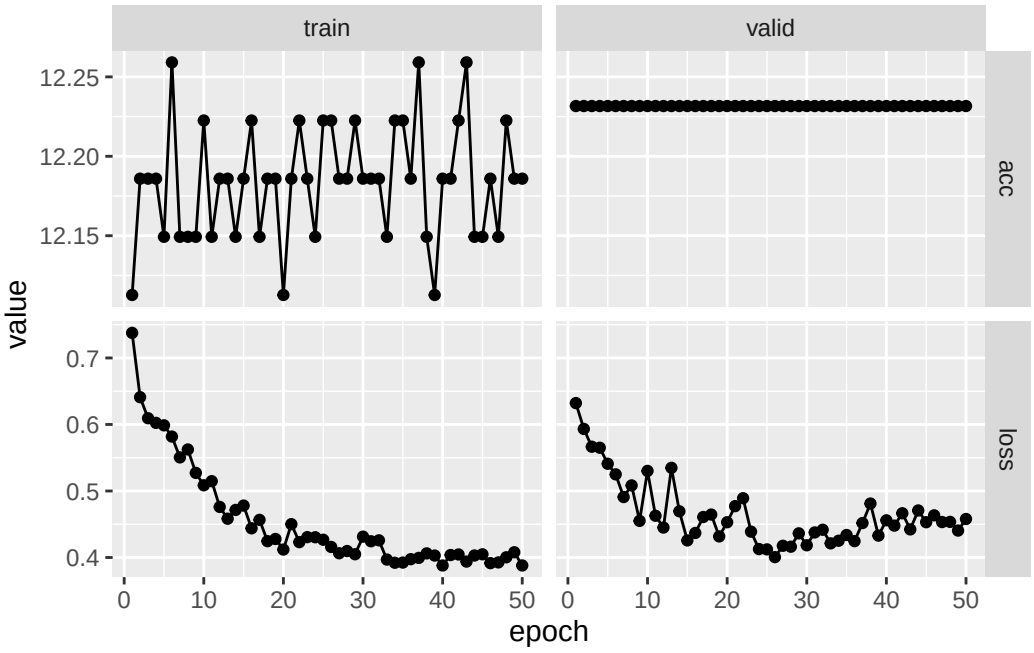
```
Epoch 1/50
Train metrics: Loss: 0.7377 - Acc: 12.1127
Valid metrics: Loss: 0.6321 - Acc: 12.2316
Epoch 2/50
Train metrics: Loss: 0.641 - Acc: 12.1859
Valid metrics: Loss: 0.5933 - Acc: 12.2316
Epoch 3/50
Train metrics: Loss: 0.6094 - Acc: 12.1859
Valid metrics: Loss: 0.5666 - Acc: 12.2316
Epoch 4/50
Train metrics: Loss: 0.6024 - Acc: 12.1859
Valid metrics: Loss: 0.565 - Acc: 12.2316
Epoch 5/50
Train metrics: Loss: 0.5987 - Acc: 12.1493
Valid metrics: Loss: 0.5409 - Acc: 12.2316
Epoch 6/50
Train metrics: Loss: 0.5818 - Acc: 12.2592
Valid metrics: Loss: 0.525 - Acc: 12.2316
Epoch 7/50
Train metrics: Loss: 0.5505 - Acc: 12.1493
Valid metrics: Loss: 0.4909 - Acc: 12.2316
```

```
Epoch 8/50
Train metrics: Loss: 0.5624 - Acc: 12.1493
Valid metrics: Loss: 0.5084 - Acc: 12.2316
Epoch 9/50
Train metrics: Loss: 0.5272 - Acc: 12.1493
Valid metrics: Loss: 0.4551 - Acc: 12.2316
Epoch 10/50
Train metrics: Loss: 0.5086 - Acc: 12.2225
Valid metrics: Loss: 0.5302 - Acc: 12.2316
Epoch 11/50
Train metrics: Loss: 0.5148 - Acc: 12.1493
Valid metrics: Loss: 0.4628 - Acc: 12.2316
Epoch 12/50
Train metrics: Loss: 0.4761 - Acc: 12.1859
Valid metrics: Loss: 0.445 - Acc: 12.2316
Epoch 13/50
Train metrics: Loss: 0.4583 - Acc: 12.1859
Valid metrics: Loss: 0.5348 - Acc: 12.2316
Epoch 14/50
Train metrics: Loss: 0.4715 - Acc: 12.1493
Valid metrics: Loss: 0.4695 - Acc: 12.2316
Epoch 15/50
Train metrics: Loss: 0.4781 - Acc: 12.1859
Valid metrics: Loss: 0.4257 - Acc: 12.2316
Epoch 16/50
Train metrics: Loss: 0.4439 - Acc: 12.2225
Valid metrics: Loss: 0.4369 - Acc: 12.2316
Epoch 17/50
Train metrics: Loss: 0.4566 - Acc: 12.1493
Valid metrics: Loss: 0.4609 - Acc: 12.2316
Epoch 18/50
Train metrics: Loss: 0.4246 - Acc: 12.1859
Valid metrics: Loss: 0.4646 - Acc: 12.2316
Epoch 19/50
Train metrics: Loss: 0.4278 - Acc: 12.1859
Valid metrics: Loss: 0.4317 - Acc: 12.2316
Epoch 20/50
Train metrics: Loss: 0.412 - Acc: 12.1127
Valid metrics: Loss: 0.453 - Acc: 12.2316
Epoch 21/50
Train metrics: Loss: 0.4501 - Acc: 12.1859
Valid metrics: Loss: 0.4772 - Acc: 12.2316
Epoch 22/50
```

```
Train metrics: Loss: 0.4231 - Acc: 12.2225
Valid metrics: Loss: 0.4891 - Acc: 12.2316
Epoch 23/50
Train metrics: Loss: 0.4306 - Acc: 12.1859
Valid metrics: Loss: 0.4389 - Acc: 12.2316
Epoch 24/50
Train metrics: Loss: 0.4305 - Acc: 12.1493
Valid metrics: Loss: 0.4127 - Acc: 12.2316
Epoch 25/50
Train metrics: Loss: 0.4267 - Acc: 12.2225
Valid metrics: Loss: 0.4122 - Acc: 12.2316
Epoch 26/50
Train metrics: Loss: 0.4159 - Acc: 12.2225
Valid metrics: Loss: 0.4006 - Acc: 12.2316
Epoch 27/50
Train metrics: Loss: 0.4064 - Acc: 12.1859
Valid metrics: Loss: 0.4176 - Acc: 12.2316
Epoch 28/50
Train metrics: Loss: 0.4097 - Acc: 12.1859
Valid metrics: Loss: 0.4163 - Acc: 12.2316
Epoch 29/50
Train metrics: Loss: 0.4049 - Acc: 12.2225
Valid metrics: Loss: 0.4363 - Acc: 12.2316
Epoch 30/50
Train metrics: Loss: 0.4314 - Acc: 12.1859
Valid metrics: Loss: 0.4184 - Acc: 12.2316
Epoch 31/50
Train metrics: Loss: 0.4244 - Acc: 12.1859
Valid metrics: Loss: 0.4377 - Acc: 12.2316
Epoch 32/50
Train metrics: Loss: 0.4258 - Acc: 12.1859
Valid metrics: Loss: 0.4417 - Acc: 12.2316
Epoch 33/50
Train metrics: Loss: 0.397 - Acc: 12.1493
Valid metrics: Loss: 0.4213 - Acc: 12.2316
Epoch 34/50
Train metrics: Loss: 0.3922 - Acc: 12.2225
Valid metrics: Loss: 0.4252 - Acc: 12.2316
Epoch 35/50
Train metrics: Loss: 0.3924 - Acc: 12.2225
Valid metrics: Loss: 0.4338 - Acc: 12.2316
Epoch 36/50
Train metrics: Loss: 0.3975 - Acc: 12.1859
```

```
Valid metrics: Loss: 0.4246 - Acc: 12.2316
Epoch 37/50
Train metrics: Loss: 0.3994 - Acc: 12.2592
Valid metrics: Loss: 0.4518 - Acc: 12.2316
Epoch 38/50
Train metrics: Loss: 0.4062 - Acc: 12.1493
Valid metrics: Loss: 0.4811 - Acc: 12.2316
Epoch 39/50
Train metrics: Loss: 0.403 - Acc: 12.1127
Valid metrics: Loss: 0.4329 - Acc: 12.2316
Epoch 40/50
Train metrics: Loss: 0.388 - Acc: 12.1859
Valid metrics: Loss: 0.4559 - Acc: 12.2316
Epoch 41/50
Train metrics: Loss: 0.4037 - Acc: 12.1859
Valid metrics: Loss: 0.4479 - Acc: 12.2316
Epoch 42/50
Train metrics: Loss: 0.4046 - Acc: 12.2225
Valid metrics: Loss: 0.4666 - Acc: 12.2316
Epoch 43/50
Train metrics: Loss: 0.3939 - Acc: 12.2592
Valid metrics: Loss: 0.4423 - Acc: 12.2316
Epoch 44/50
Train metrics: Loss: 0.403 - Acc: 12.1493
Valid metrics: Loss: 0.4708 - Acc: 12.2316
Epoch 45/50
Train metrics: Loss: 0.4048 - Acc: 12.1493
Valid metrics: Loss: 0.4531 - Acc: 12.2316
Epoch 46/50
Train metrics: Loss: 0.3916 - Acc: 12.1859
Valid metrics: Loss: 0.4634 - Acc: 12.2316
Epoch 47/50
Train metrics: Loss: 0.3926 - Acc: 12.1493
Valid metrics: Loss: 0.4533 - Acc: 12.2316
Epoch 48/50
Train metrics: Loss: 0.4003 - Acc: 12.2225
Valid metrics: Loss: 0.4536 - Acc: 12.2316
Epoch 49/50
Train metrics: Loss: 0.4079 - Acc: 12.1859
Valid metrics: Loss: 0.4405 - Acc: 12.2316
Epoch 50/50
Train metrics: Loss: 0.3881 - Acc: 12.1859
Valid metrics: Loss: 0.4579 - Acc: 12.2316
```

```
plot(fit_nn)
```




```
nn_test <- predict(
    fit_nn,
    model.matrix(y ~ . - 1, data = df_test)
)
# nn_test
nn_preds <- ifelse(nn_test > 0.5, 1, 0)

table(nn_preds, df_test$y)
```

```
nn_preds  0  1
       0 98 29
       1  9 41
```

```
mean(nn_preds == df_test$y)
```

```
[1] 0.7853107
```

```
table(glm_preds, df_test$y)
```

```
glm_preds  0  1
        0 98 26
        1  9 44
```

```
mean(glm_preds == df_test$y)
```

```
[1] 0.8022599
```

DataLoaders

- key component in the ML pipeline
- handle loading and preprocessing of data in efficient way for training and evaluating models
- make it easy to work with large datasets (ex. dataset with trillions of bits)
  - Loading data in smaller batches called chunks
- Advantages

1. Efficiency memory management
2. Parallelism
3. Preprocessing
4. Flexibility
5. Standardization

- involves partitioning data into smaller chunks (batches)
- inside one large epoch of gradient descent, it'll break up the data into batches and mini gradient descents (does gradient descent on every batch)
- IS DIFFERENT FROM CROSS VALIDATION
- training data is split up into smaller chunks - data is being cycled through during gradient descent
- if batches are fixed statically (first 100 are batch 1, next 100 are batch 2), this is going to introduce some bias into the gradient descent process, so you want to shuffle the data between every epoch
- DataLoader takes care of these issues

```
dir <- "./mnist"

train_ds <- mnist_dataset(
```

```
    root = dir,
    train = TRUE,
    download = TRUE,
    transform = transform
)
test_ds <- mnist_dataset(
    root = dir,
    train = FALSE,
    download = TRUE,
    transform = transform
)
```

```
typeof(train_ds)
```

```
[1] "environment"
```

```
length(train_ds)
```

```
[1] 60000
```

```
# 42,000nd  observation is a 28x28 matrix
# one row is a 28x28 matrix
# is an image where a value specifies the intensity of the pixels in the 28x28 image
train_ds$data[42000, ,]
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
 [1,]    0    0    0    0    0    0    0    0    0     0     0     0     0
 [2,]    0    0    0    0    0    0    0    0    0     0     0     0     0
 [3,]    0    0    0    0    0    0    0    0    0     0     0     0     0
 [4,]    0    0    0    0    0    0    0    0    0     0     0     0     0
 [5,]    0    0    0    0    0    0    0    0    0     0     0     0     0
 [6,]    0    0    0    0    0    0    0    0    0     0     0     0     0
 [7,]    0    0    0    0    0    0    0    0    0     0     0     6    37
 [8,]    0    0    0    0    0    0    0    0    0     0   125   160   252
 [9,]    0    0    0    0    0    0    0    0    1   109   232   252   252
[10,]    0    0    0    0    0    0    0    0  125   252   252   252   252
[11,]    0    0    0    0    0    0    0    0   62   189   211   252   252
[12,]    0    0    0    0    0    0    0   21  206   252   190   252   168
```

```
[13,]   0   0   0   0   0   0  73 253 253 253 253 217   0
[14,]   0   0   0   0   0   0 115 252 252 252 148  30   0
[15,]   0   0   0   0   0   0 217 252 252 252  35   0   0
[16,]   0   0   0   0   0   0 217 252 252 252  35   0   0
[17,]   0   0   0   0   0 110 233 253 253 144   0  79 109
[18,]   0   0   0   0   0 253 252 252 252 237 217 242 252
[19,]   0   0   0   0   0 253 252 252 252 252 252 252 252
[20,]   0   0   0   0   0 170 252 252 252 252 252 252 252
[21,]   0   0   0   0   0   0 218 253 253 253 253 253 253
[22,]   0   0   0   0   0   0  72 231 252 252 252 252 252
[23,]   0   0   0   0   0   0   0  52  71  71  71  71  71
[24,]   0   0   0   0   0   0   0   0   0   0   0   0   0
[25,]   0   0   0   0   0   0   0   0   0   0   0   0   0
[26,]   0   0   0   0   0   0   0   0   0   0   0   0   0
[27,]   0   0   0   0   0   0   0   0   0   0   0   0   0
[28,]   0   0   0   0   0   0   0   0   0   0   0   0   0
      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
 [1,]     0     0     0     0     0     0     0     0     0     0     0     0
 [2,]     0     0     0     0     0     0     0     0     0     0     0     0
 [3,]     0     0     0     0     0     0     0     0     0     0     0     0
 [4,]     0     0     0     0     0     0     0     0     0     0     0     0
 [5,]     0     0     0     0     0     0     0     0     0     0     0     0
 [6,]     0     0     0     0     0     0     0    42   218   134   186     0
 [7,]   182    98    51     0     0     0    27   221   253   252   221    16
 [8,]   253   252   175   144     0     0    16   190   253   252   252   108
 [9,]   253   252   252   252     0     0     0     0   109   252   236    62
[10,]   253   252   200   179     0     0     0     0   109   252   215    42
[11,]   237    91    20     0     0     0     0    21   212   252   241   221
[12,]    62     0     0     0     0     0    21   206   253   252   252   252
[13,]     0     0     0     0     0    32   212   253   255   253   253   108
[14,]     0     0     0     0     0   115   252   252   253   252   220    15
[15,]     0     0    27   120   182   242   252   252   253   252   112     0
[16,]     0   125   221   252   253   252   252   252   253   128    31     0
[17,]   255   253   253   253   255   253   253   253   208    20     0     0
[18,]   253   252   252   252   253   252   252   210    20     0     0     0
[19,]   253   252   252   252   217   215   112    31     0     0     0     0
[20,]   253   252   252   252     0     0     0     0     0     0     0     0
[21,]   255   253   175    62     0     0     0     0     0     0     0     0
[22,]   119    35    10     0     0     0     0     0     0     0     0     0
[23,]     0     0     0     0     0     0     0     0     0     0     0     0
[24,]     0     0     0     0     0     0     0     0     0     0     0     0
[25,]     0     0     0     0     0     0     0     0     0     0     0     0
[26,]     0     0     0     0     0     0     0     0     0     0     0     0
```

```
[27,]    0    0    0    0    0    0    0    0    0    0    0    0
[28,]    0    0    0    0    0    0    0    0    0    0    0    0
      [,26] [,27] [,28]
 [1,]     0     0     0
 [2,]     0     0     0
 [3,]     0     0     0
 [4,]     0     0     0
 [5,]     0     0     0
 [6,]     0     0     0
 [7,]     0     0     0
 [8,]     0     0     0
 [9,]     0     0     0
[10,]     0     0     0
[11,]     0     0     0
[12,]     0     0     0
[13,]     0     0     0
[14,]     0     0     0
[15,]     0     0     0
[16,]     0     0     0
[17,]     0     0     0
[18,]     0     0     0
[19,]     0     0     0
[20,]     0     0     0
[21,]     0     0     0
[22,]     0     0     0
[23,]     0     0     0
[24,]     0     0     0
[25,]     0     0     0
[26,]     0     0     0
[27,]     0     0     0
[28,]     0     0     0
```

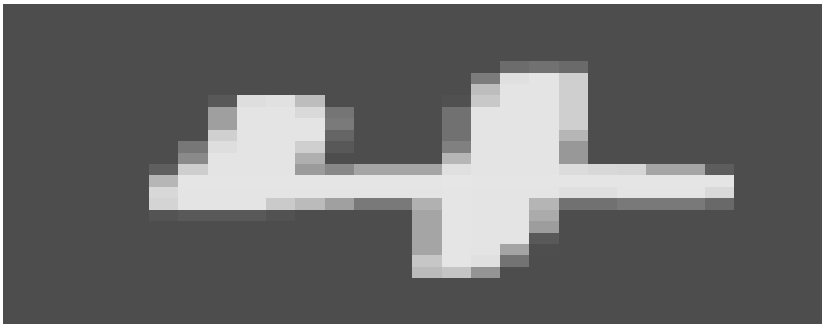
Using the data/DataLoaders to determine the value of a handwritten number

```
# collection of {x_i, y_i} where i = 1, ranging to 10000
# x_i is 28x28 image of handwritten image
# every y_i {0,1,...,9}
options(repr.plot.width=10, repr.plot.height=10)

i <- sample(1:length(train_ds), 1)
x <- train_ds$data[i, ,] %>% t
```

```r
image(x[1:28, 28:1], useRaster=TRUE, axes=FALSE, col=gray.colors(1000), main = train_ds$ta
```

4



- data frame with each row being a single image observation and its actual value
- p (# cols) = # of pixels in width^2 so you can have column for each pixel

```r
# splitting up batches into size 128
# shuffling allows to negate any bias by shuffling observations for each epoch
train_dl <- dataloader(train_ds, batch_size = 1024, shuffle = TRUE)
test_dl <- dataloader(test_ds, batch_size = 1024)
```

```r
NNet_10 <- nn_module(
  initialize = function(p, q1, q2, q3, o) {
    self$hidden1 <- nn_linear(p, q1)
    self$hidden2 <- nn_linear(q1, q2)
    self$hidden3 <- nn_linear(q2, q3)
    self$OUTPUT <- nn_linear(q3, o)
    self$activation <- nn_relu()
  },
  forward = function(x) {
    x %>%
      self$hidden1() %>%
```

```
      self$activation() %>%
      self$hidden2() %>%
      self$activation() %>%
      self$hidden3() %>%
      self$activation() %>%
      self$OUTPUT()
  }
)
```

Unable to get this chunk to work as it gives error that says it can't convert argument: input

```
fit_nn <- NNet_10 %>%
    #
    # Setup the model
    #
    setup(
        loss = nn_cross_entropy_loss(),
        optimizer = optim_adam,
        metrics = list(
            luz_metric_accuracy()
        )
    ) %>%
    #
    # Set the hyperparameters
    #
    set_hparams(p=28*28, q1=256, q2=128, q3=64, o=10) %>%
    #
    # Fit the model
    #
    fit(
        epochs = 10,
        data = train_dl,
        valid_data = test_dl,
        verbose=TRUE
    )

NN10_preds <- fit_nn %>%
  predict(test_ds) %>%
  torch_argmax(dim = 2) %>%
  as_array()
```

Thursday, April 13

! TIL

Include a *very brief* summary of what you learnt in this class here.
Today, I learnt the following concepts in class:

1. Item 1
2. Item 2
3. Item 3