Weekly Summary Template

Brady Miller

Table of contents

Tuesday, April 18 6
Thursday, April 20 11

```r
packages <- c(
    # Old packages
    "ISLR2",
    "dplyr",
    "tidyr",
    "readr",
    "purrr",
    "repr",
    "tidyverse",
    "kableExtra",
    "IRdisplay",
    # NEW
    "torch",
    "torchvision",
    "luz",
    "dimRed",
    "RSpectra",
    'corrplot',
    'car'
)

renv::install(packages)
```

```
Installing ISLR2 [1.3-2] ...
    OK [linked cache in 2.9 milliseconds]
```

```
Installing dplyr [1.1.1] ...
    OK [linked cache in 3.9 milliseconds]
Installing purrr [1.0.1] ...
    OK [linked cache in 2.4 milliseconds]
Installing tidyr [1.3.0] ...
    OK [linked cache in 3.6 milliseconds]
Installing readr [2.1.4] ...
    OK [linked cache in 2.7 milliseconds]
Installing repr [1.1.6] ...
    OK [linked cache in 3.2 milliseconds]
Installing tidyverse [2.0.0] ...
    OK [linked cache in 3.7 milliseconds]
Installing kableExtra [1.3.4] ...
    OK [linked cache in 3.6 milliseconds]
Installing IRdisplay [1.1] ...
    OK [linked cache in 2.6 milliseconds]
Installing torch [0.10.0] ...
    OK [linked cache in 2.6 milliseconds]
Installing torchvision [0.5.1] ...
    OK [linked cache in 3.9 milliseconds]
Installing luz [0.4.0] ...
    OK [linked cache in 2.9 milliseconds]
Installing dimRed [0.2.6] ...
    OK [linked cache in 2.9 milliseconds]
Installing RSpectra [0.16-1] ...
    OK [linked cache in 3.4 milliseconds]
Installing corrplot [0.92] ...
    OK [linked cache in 3 milliseconds]
Installing car [3.1-2] ...
    OK [linked cache in 3.6 milliseconds]
```

```
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

-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
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
```

```
Loading required package: dimRed

Loading required package: DRR

Loading required package: kernlab


Attaching package: 'kernlab'


The following object is masked from 'package:ggplot2':

    alpha


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

    cross


Loading required package: CVST

Loading required package: Matrix


Attaching package: 'Matrix'


The following objects are masked from 'package:tidyr':

    expand, pack, unpack



Attaching package: 'dimRed'


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

    embed


The following object is masked from 'package:base':
```

```
    as.data.frame


Loading required package: RSpectra

Loading required package: corrplot

corrplot 0.92 loaded

Loading required package: car

Loading required package: carData


Attaching package: 'car'


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

    some


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

    recode

      ISLR2       dplyr       tidyr       readr       purrr        repr
       TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
  tidyverse  kableExtra   IRdisplay       torch torchvision         luz
       TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
     dimRed    RSpectra    corrplot         car
       TRUE        TRUE        TRUE        TRUE
```

Tuesday, April 18

> **!** TIL
>
> Include a *very brief* summary of what you learnt in this class here.
> Today, I learnt the following concepts in class:
>
> 1. Finding PCAs
> 2. Factor loadings and their meaning
> 3. Item 3

Finding PCAs

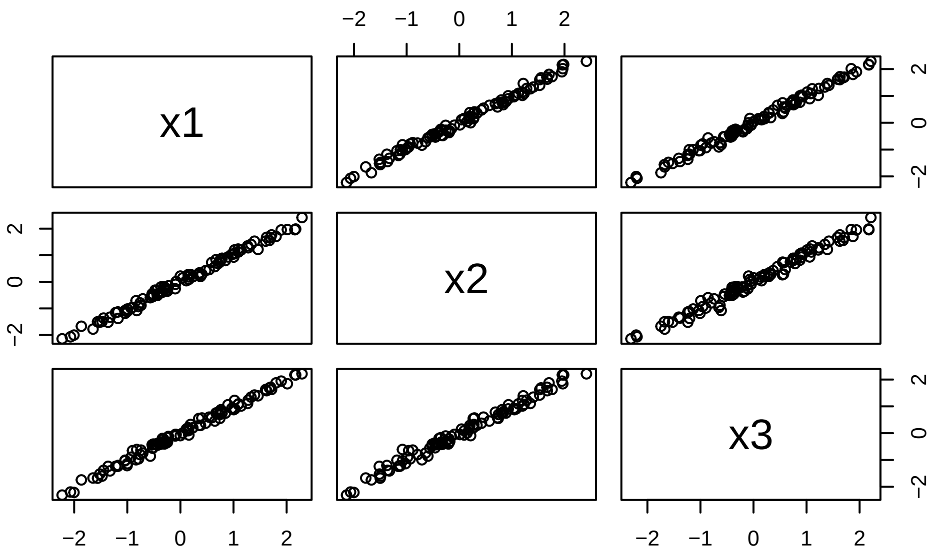
```r
data <- tibble(
  x1 = rnorm(100, mean = 0, sd = 1),
  x2 = x1 + rnorm(100, mean = 0, sd = 0.1),
  x3 = x1 + rnorm(100, mean = 0, sd = 0.1)
)

head(data) %>% knitr::kable()
```

x1	x2	x3
-0.5214967	-0.5600559	-0.5633774
0.3785114	0.2001956	0.2881111
-0.7878213	-0.9328268	-0.9590303
1.6984192	1.6668636	1.7109405
0.5893024	0.7273202	0.5757653
0.1904472	0.2762988	0.3237107

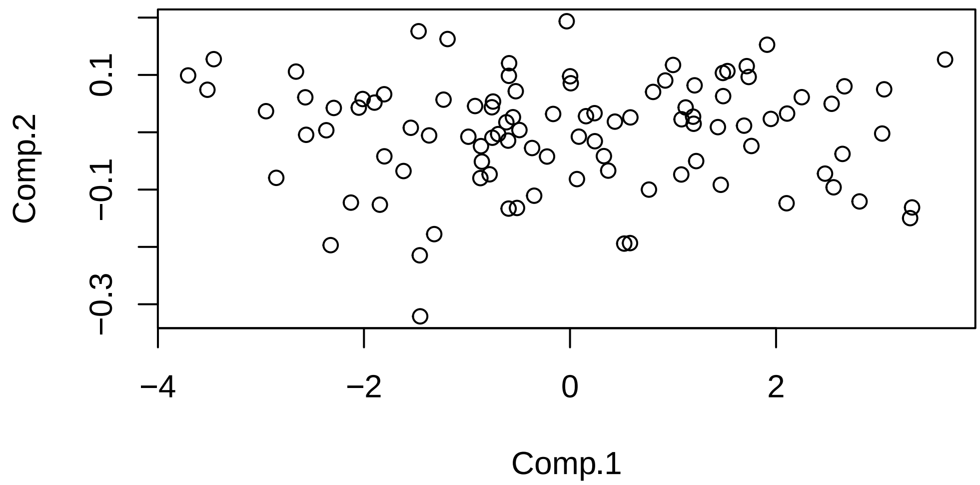
```r
pca <- princomp(data, cor = TRUE)
summary(pca)
```

```
Importance of components:
                          Comp.1      Comp.2      Comp.3
Standard deviation     1.7285462 0.094464136 0.056607920
Proportion of Variance 0.9959574 0.002974491 0.001068152
Cumulative Proportion  0.9959574 0.998931848 1.000000000
```

```
par(mfrow=c(1, 2))

Z_pca <- predict(pca, data)
plot(data)
```




```
plot(Z_pca)
```



Finding factor loadings and their meanings

```
pca$loadings
```

```
Loadings:
   Comp.1 Comp.2 Comp.3
x1  0.578         0.816
x2  0.577  0.704 -0.414
x3  0.577 -0.710 -0.403

               Comp.1 Comp.2 Comp.3
SS loadings     1.000  1.000  1.000
Proportion Var  0.333  0.333  0.333
Cumulative Var  0.333  0.667  1.000
```

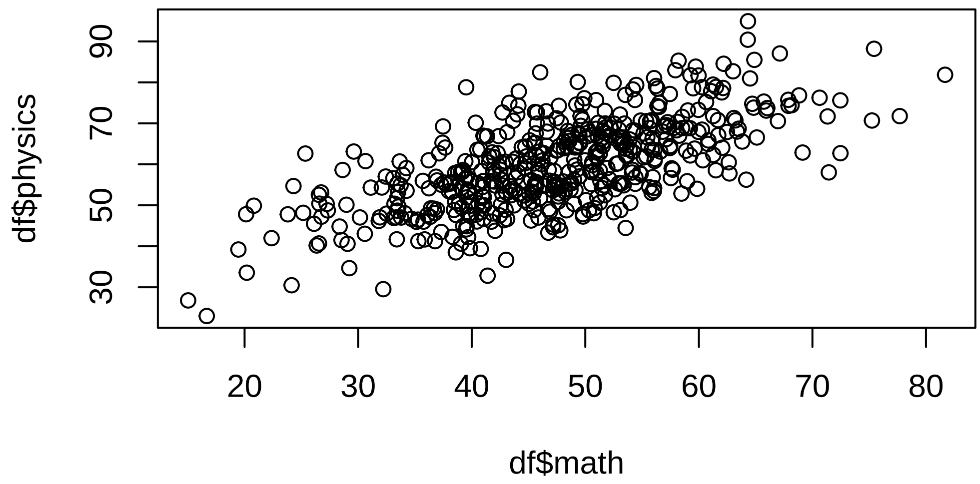
Example of finding PCAs using test score data

```
set.seed(42)

n <- 500
```

```r
science <- rnorm(n, mean = 60, sd = 10)
humanities <- rnorm(n, mean = 80, sd=10)

df <- tibble(
  math = 0.8 * science + rnorm(n, mean = 0, sd = 7),
  physics = 1.0 * science + rnorm(n, mean = 0, sd = 5),
  chemistry = 1.3 * science + rnorm(n, mean = 0, sd = 3),
  history = 0.8 * humanities + rnorm(n, mean = 0, sd = 5),
  geography = 1.0 * humanities + rnorm(n, mean = 0, sd = 10),
  literature = 1.2 * humanities + rnorm(n, mean = 0, sd = 2)
)

df %>%
    head() %>%
    round(digits = 2) %>%
    knitr::kable()
```

math	physics	chemistry	history	geography	literature
75.24	70.70	96.57	75.32	83.43	108.93
47.15	53.67	69.83	71.30	81.22	104.37
57.70	58.69	77.55	63.52	75.91	95.74
55.70	70.49	80.21	67.09	69.87	97.45
44.26	60.07	79.38	61.18	83.96	87.11
42.97	60.64	77.72	62.33	69.22	91.86

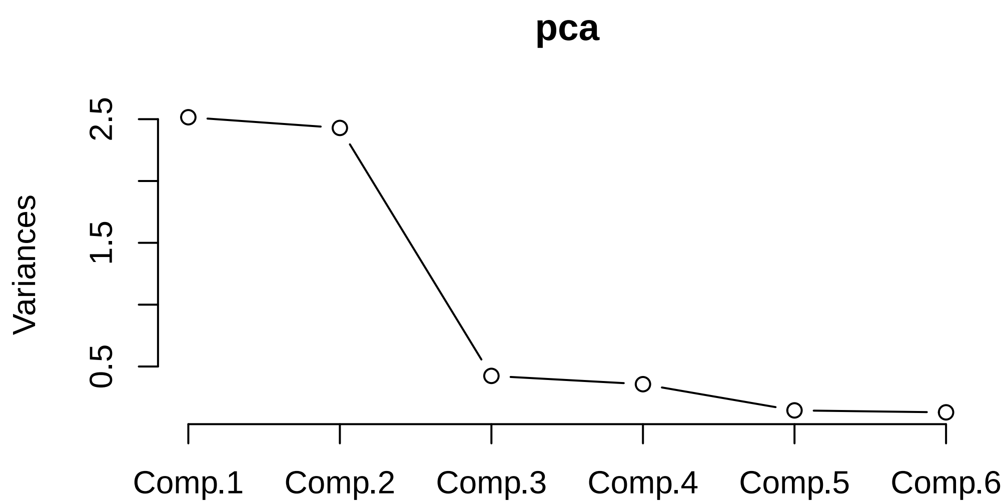
```r
plot(df$math, df$physics)
```




```r
pca <- princomp(df, cor=TRUE)
summary(pca)
```

```
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4     Comp.5
Standard deviation     1.5860772 1.5585727 0.65113393 0.59751444 0.38083707
Proportion of Variance 0.4192735 0.4048581 0.07066257 0.05950392 0.02417281
Cumulative Proportion  0.4192735 0.8241316 0.89479419 0.95429811 0.97847093
                           Comp.6
Standard deviation     0.35940847
Proportion of Variance 0.02152907
Cumulative Proportion  1.00000000
```

```r
plot(pca, type="l")
```



Thursday, April 20

! TIL

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

1. Principal component regression using test score data
2. Nonlinear dimension reduction
3. Autoencoder/decoder process

Principal component regression using test score data

Principal component regression

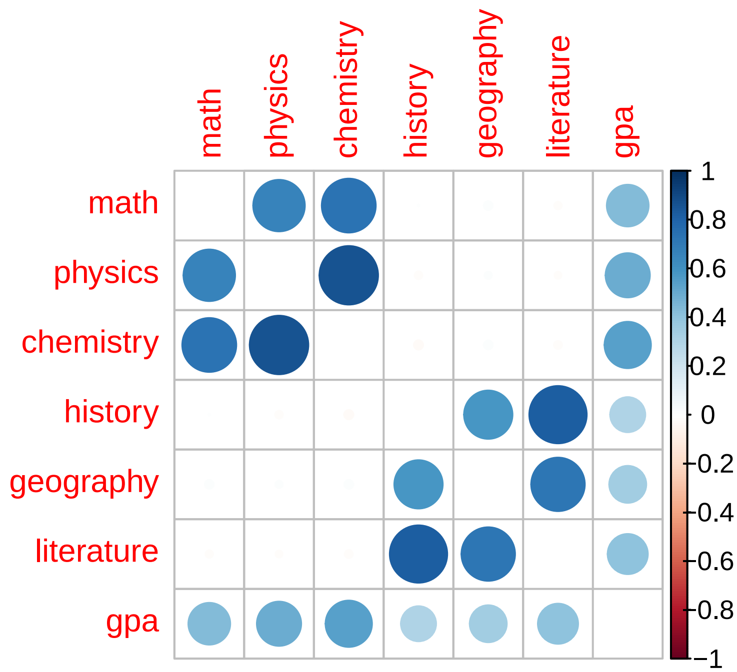
```
df$gpa <- (0.9 * science + 0.5 * humanities + rnorm(n, mean=0, sd=10)) * 4 / 100

df %>%
    head() %>%
    round(digits=2) %>%
    knitr::kable()
```

math	physics	chemistry	history	geography	literature	gpa
75.24	70.70	96.57	75.32	83.43	108.93	4.40
47.15	53.67	69.83	71.30	81.22	104.37	3.41
57.70	58.69	77.55	63.52	75.91	95.74	3.76
55.70	70.49	80.21	67.09	69.87	97.45	4.17
44.26	60.07	79.38	61.18	83.96	87.11	2.96
42.97	60.64	77.72	62.33	69.22	91.86	3.28

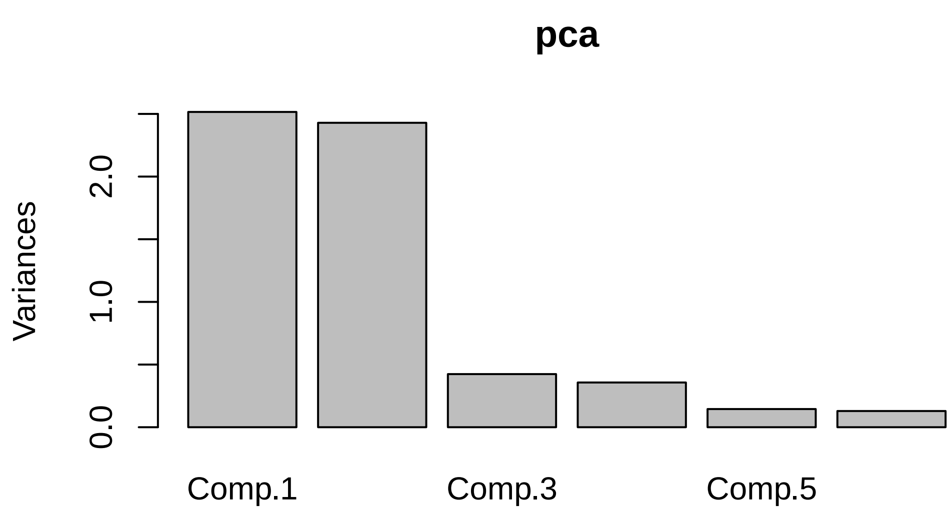
```r
lm_fit <- lm(gpa ~ ., df)
summary(lm_fit)
```

```
Call:
lm(formula = gpa ~ ., data = df)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.12706 -0.29370  0.01835  0.28236  1.25832 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.017263   0.187783  -0.092   0.9268    
math         0.004944   0.002671   1.851   0.0647 .  
physics      0.004557   0.003467   1.315   0.1892    
chemistry    0.018271   0.003217   5.680 2.31e-08 ***
history     -0.004430   0.003469  -1.277   0.2022    
geography    0.002817   0.001873   1.504   0.1332    
literature   0.019673   0.003171   6.205 1.16e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4241 on 493 degrees of freedom
Multiple R-squared:  0.4736,	Adjusted R-squared:  0.4671 
F-statistic: 73.91 on 6 and 493 DF,  p-value: < 2.2e-16
```

```r
df %>%
    cor() %>%
    corrplot(diag=F)
```




```
pca <- princomp(df %>% select(-gpa), cor=TRUE)
screeplot(pca)
```

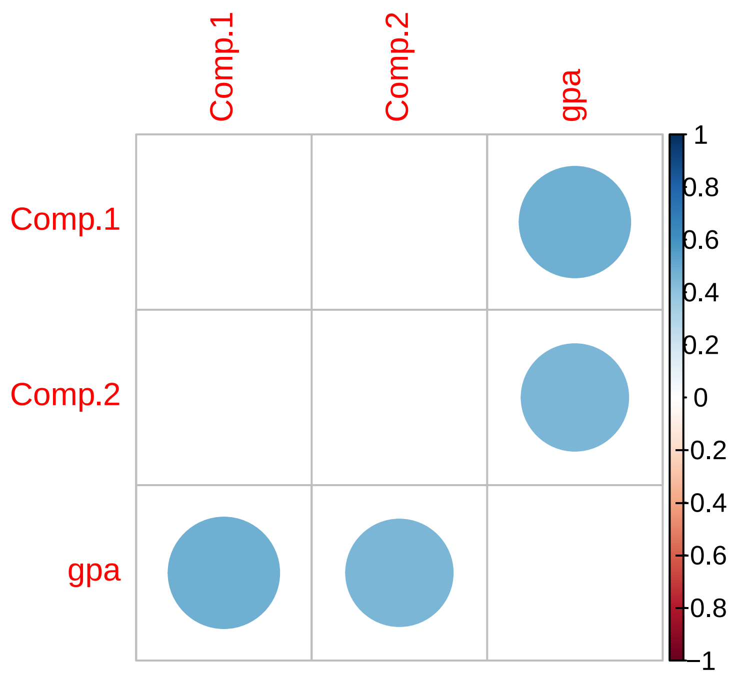


```r
Z <- predict(pca, df)
```

```r
df_pca <- Z %>%
    as_tibble %>%
    select(Comp.1, Comp.2) %>%
    mutate(gpa = df$gpa)

head(df_pca) %>% knitr::kable()
```

Comp.1	Comp.2	gpa
2.7037738	1.8078753	4.402650
-0.8243280	0.8429457	3.414104
0.4763584	-0.0850971	3.760019
1.1033923	0.0555130	4.166295
-0.0155603	-0.4092377	2.963712
-0.1197505	-0.6725353	3.282428

```r
df_pca %>%
    cor() %>%
    corrplot(diag=F)
```



```r
lm_pca <- lm(gpa ~ ., df_pca)
summary(lm_pca)
```

```
Call:
lm(formula = gpa ~ ., data = df_pca)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.22601 -0.30774  0.01379  0.28813  1.25162 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.73356    0.01940  192.43   <2e-16 ***
Comp.1       0.17880    0.01223   14.62   <2e-16 ***
Comp.2       0.16893    0.01245   13.57   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4338 on 497 degrees of freedom
Multiple R-squared:  0.4446,	Adjusted R-squared:  0.4423 
F-statistic: 198.9 on 2 and 497 DF,  p-value: < 2.2e-16
```

```r
vif(lm_pca) %>% t
```

```
     Comp.1 Comp.2
[1,]      1      1
```

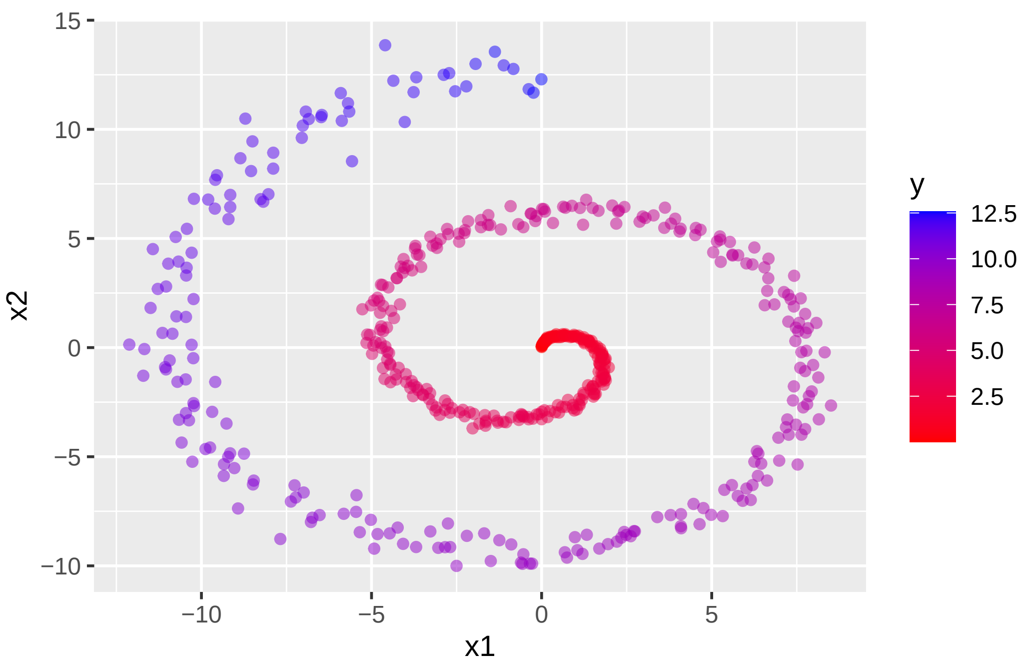
Nonlinear dimension reduction

```r
generate_two_spirals <- function(){
  set.seed(42)
  n <- 500
  noise <- 0.05
  t <- (1:n) / n * 4 * pi
  x1 <- t * (sin(t) + rnorm(n, 0, noise))
  x2 <- t * (cos(t) + rnorm(n, 0, noise))
  y  <- t
  return(tibble(x1=x1, x2=x2, y=y))
}
```

```
df <- generate_two_spirals()
head(df)
```

```
# A tibble: 6 x 3
       x1     x2      y
    <dbl>  <dbl>  <dbl>
1 0.00235 0.0264 0.0251
2 0.00111 0.0525 0.0503
3 0.00705 0.0752 0.0754
4 0.0133  0.101  0.101
5 0.0183  0.120  0.126
6 0.0219  0.148  0.151
```

```
ggplot(df) +
    geom_point(aes(x=x1, y=x2, col=y), alpha=0.5) +
    scale_colour_gradient(low="red", high="blue")
```



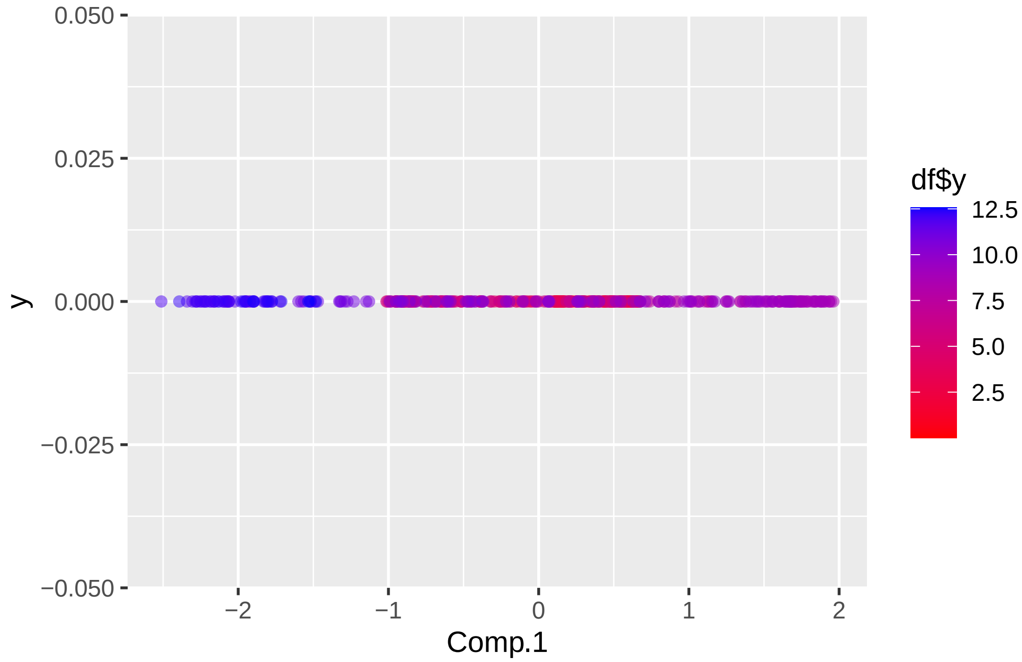

```
pca <- princomp(df[, 1:2], cor=T)
pca$loadings
```

```
Loadings:
   Comp.1 Comp.2
x1  0.707  0.707
x2 -0.707  0.707

               Comp.1 Comp.2
SS loadings       1.0    1.0
Proportion Var    0.5    0.5
Cumulative Var    0.5    1.0
```

```r
df_pca <- predict(pca, df)
head(df_pca)
```

```
        Comp.1    Comp.2
[1,] 0.1410355 0.1445642
[2,] 0.1373030 0.1479455
[3,] 0.1350473 0.1518737
[4,] 0.1324411 0.1562312
[5,] 0.1304964 0.1595888
[6,] 0.1272554 0.1638327
```

```r
ggplot(as_tibble(df_pca)) +
    geom_point(aes(x=Comp.1, y=0, col=df$y), alpha=0.5) +
    scale_colour_gradient(low="red", high="blue")
```



Autoencoders

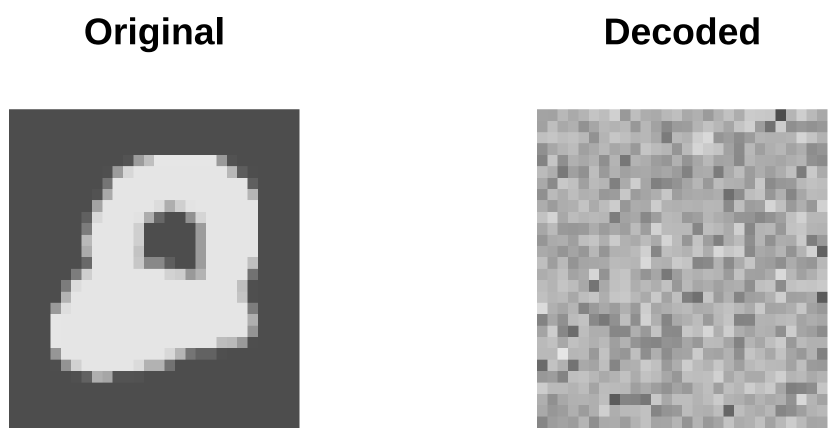
```
autoencoder <- nn_module(
    initialize = function(p, q1, q2, q3, o) {
    self$encoder <- nn_sequential(
        nn_linear(p, q1), nn_relu(),
        nn_linear(q1, q2), nn_relu(),
        nn_linear(q2, q3), nn_relu(),
        nn_linear(q3, o)
    )
    self$decoder <- nn_sequential(
        nn_linear(o, q3), nn_relu(),
        nn_linear(q3, q2), nn_relu(),
        nn_linear(q2, q1), nn_relu(),
        nn_linear(q1, p)
    )
    },
    forward = function(x) {
    x %>%
        torch_reshape(c(-1, 28 * 28)) %>%
        self$encoder() %>%
```

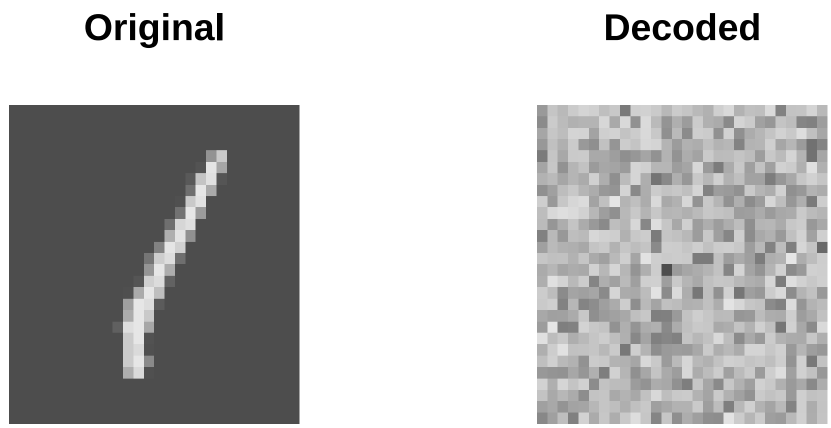
```
        self$decoder() %>%
        torch_reshape(c(-1, 28, 28))
    },
    predict = function(x) {
    x %>%
        torch_reshape(c(-1, 28 * 28)) %>%
        self$encoder()
    }
)

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
)

X <- test_ds
inputs <- torch_tensor(X$data * 1.0)

plot_image = \(x) image(t(x)[1:28, 28:1], useRaster=TRUE, axes=FALSE, col=gray.colors(1000
```

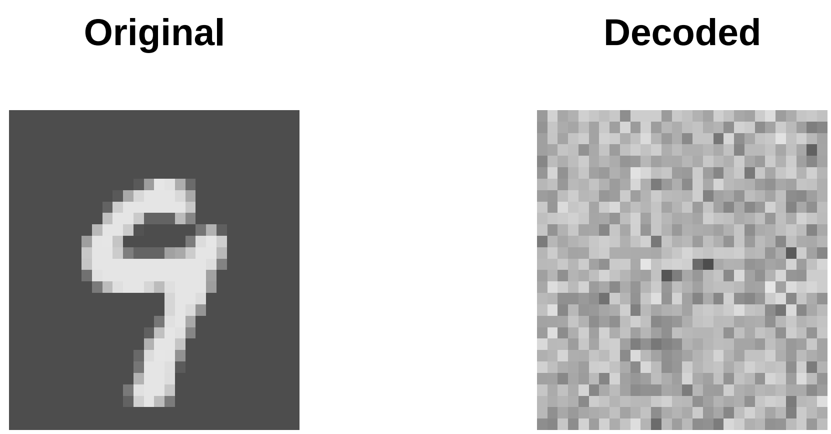
Original vs. Decoded (at initialization)

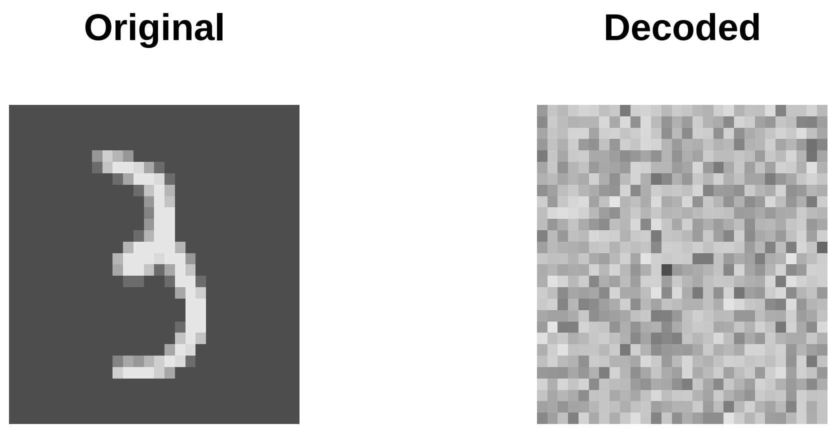
```
AE <- autoencoder(p = 28 * 28, q1 = 32, q2 = 16, q3 = 8, o = 2)

par(mfrow=c(4, 2))

set.seed(123)
for(k in 1:4){
    i <- sample(1:10000, 1)
    input <- inputs[i]
    output <- AE(inputs[i:i])[1]
```

```
    par(mfrow=c(1, 2))
    plot_image(inputs[i] %>% as_array)
    title("Original")

    plot_image(output %>% as_array)
    title("Decoded")
}
```




Original
Decoded


Original
Decoded



Fitting the autoencoder using luz

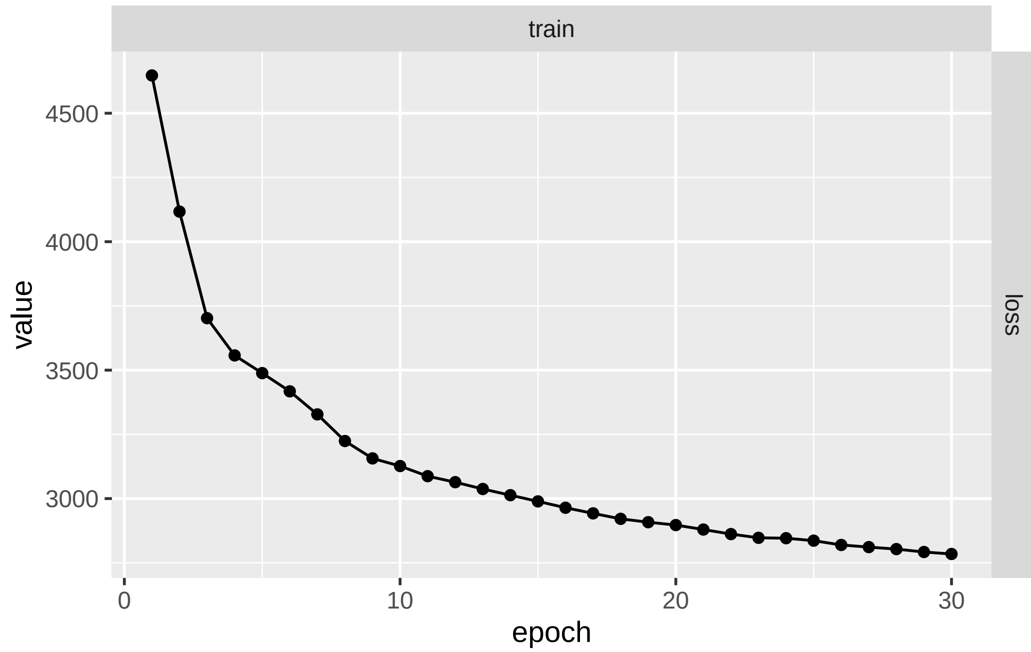
```
ae_fit <- autoencoder %>%
    setup(
        loss = nn_mse_loss(),
        optimizer = optim_adam
    ) %>%

    set_hparams(
        p=28*28, q1=128, q2=64, q3=32, o=2
    ) %>%

    set_opt_hparams(
        lr=1e-3
    ) %>%

    fit(
        data = list(
            inputs,
            inputs # targets are the same as inputs
        ),
        epochs=30,
```

```
          verbose=TRUE,
          dataloader_options = list(
              batch_size = 100,
              shuffle=TRUE
          ),
          callbacks = list(
              luz_callback_lr_scheduler(
                  torch::lr_step,
                  step_size = 10,
                  gamma=1.01
                  )
          )
      )
```

```
Epoch 1/30
Train metrics: Loss: 4647.2012
Epoch 2/30
Train metrics: Loss: 4117.1089
Epoch 3/30
Train metrics: Loss: 3702.2661
Epoch 4/30
Train metrics: Loss: 3557.3
Epoch 5/30
Train metrics: Loss: 3488.3352
Epoch 6/30
Train metrics: Loss: 3417.4993
Epoch 7/30
Train metrics: Loss: 3327.7737
Epoch 8/30
Train metrics: Loss: 3224.0479
Epoch 9/30
Train metrics: Loss: 3156.4226
Epoch 10/30
Train metrics: Loss: 3126.5886
Epoch 11/30
Train metrics: Loss: 3087.2634
Epoch 12/30
Train metrics: Loss: 3063.8831
Epoch 13/30
Train metrics: Loss: 3037.3062
Epoch 14/30
Train metrics: Loss: 3013.2427
```

```
Epoch 15/30
Train metrics: Loss: 2989.1013
Epoch 16/30
Train metrics: Loss: 2964.3701
Epoch 17/30
Train metrics: Loss: 2942.4131
Epoch 18/30
Train metrics: Loss: 2921.1714
Epoch 19/30
Train metrics: Loss: 2908.0918
Epoch 20/30
Train metrics: Loss: 2896.9031
Epoch 21/30
Train metrics: Loss: 2879.4778
Epoch 22/30
Train metrics: Loss: 2861.8184
Epoch 23/30
Train metrics: Loss: 2847.2781
Epoch 24/30
Train metrics: Loss: 2845.7949
Epoch 25/30
Train metrics: Loss: 2836.1653
Epoch 26/30
Train metrics: Loss: 2819.4712
Epoch 27/30
Train metrics: Loss: 2811.1335
Epoch 28/30
Train metrics: Loss: 2803.3757
Epoch 29/30
Train metrics: Loss: 2791.9932
Epoch 30/30
Train metrics: Loss: 2784.3438
```

```
plot(ae_fit)
```

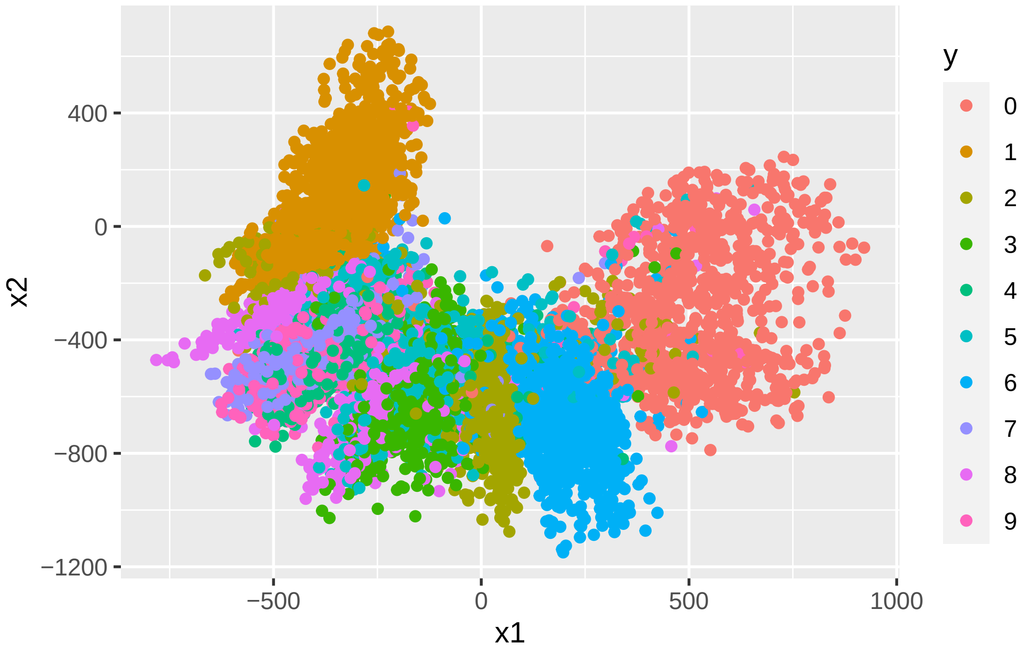


Lower-dimensional Encoding of the Data

```
X_dim2 <- predict(ae_fit, inputs) %>% as_array()
head(X_dim2)
```

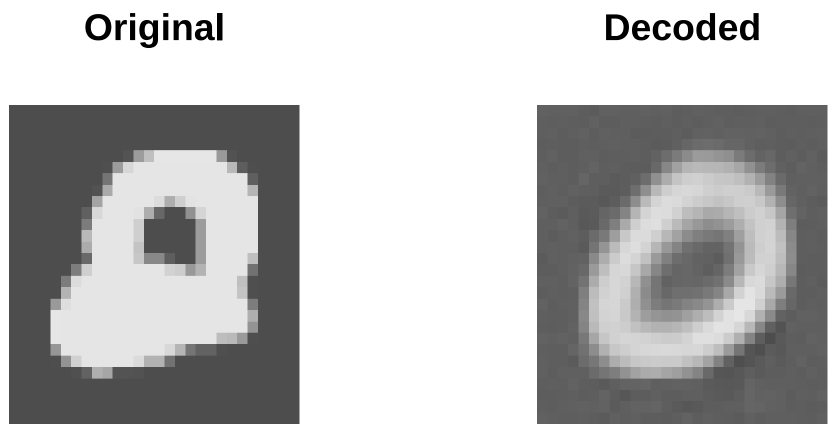
```
           [,1]       [,2]
[1,] -348.33856 -353.91779
[2,]  -72.63109 -481.36859
[3,] -284.59756   37.00065
[4,]  539.96985 -437.61911
[5,] -253.25035 -385.93155
[6,] -373.02310   33.02436
```

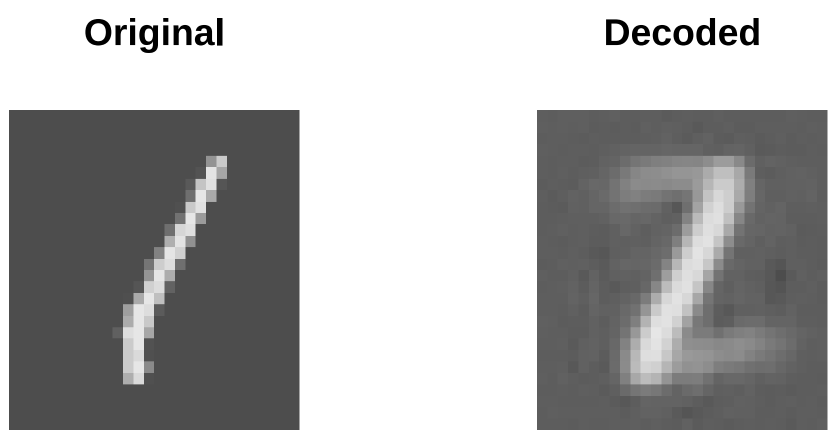
```
df_ae <- tibble(
    x1 = X_dim2[, 1],
    x2 = X_dim2[, 2],
    y = as.factor(X$targets - 1)
)

ggplot(df_ae) +
    geom_point(aes(x=x1, y=x2, col=y))
```

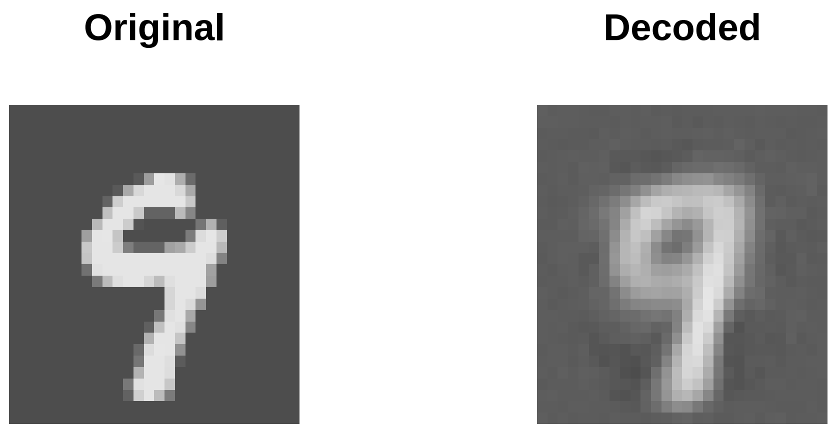


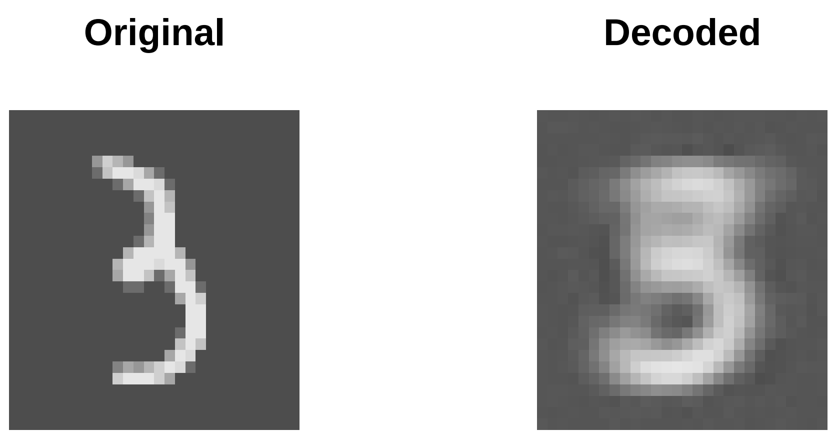
Original vs. Decoded (after fitting)

```
par(mfrow=c(4, 2))

set.seed(123)
for(k in 1:4){
    i <- sample(1:10000, 1)
    input <- inputs[i]
    output <- ae_fit$model$forward(inputs[i:i])[1]

    par(mfrow=c(1, 2))
    plot_image(inputs[i] %>% as_array)
    title("Original")

    plot_image(output %>% as_array)
    title("Decoded")
}
```


Original
Decoded


Original
Decoded


Original
Decoded


Original
Decoded