GET STARTED

- Guess the correlation
- Get the dataset
- Work with matrices
- Create the recipe
- Train the network
- Evaluate performance

USE TIDYCLS

- Get the dataset
- Work with matrices
- Create the recipe
- Train the network
- Evaluate performance

1 Guess the correlation

Get the packages

To use tidycls, you first need to install it. Install it from its high-level wrapper, tidy. While both tidy all the tidyverse functionality, tidy provides a declarative, concise API that lets you train a network in a few lines of code.

To get the respective CRAN versions, do:

```
install.packages("tidy")
install.packages("tidycls")
```

Does it work? Here's a quick test:

```
library(tidy)
library(tidycls)
data(mtcars)
fit <- mtcars %>%
  tidycls::mlp(
    .formula = mpg ~ .)
```

Now, while tidycls contains all the core functionality, and tidy is the training logic, there is a whole ecosystem built around them.

Notably, tidycls/plot is essential to image-processing tasks. In the example, we don't use it much—mainly, the U-Net uses it to export intermediate tensors for review. Let's get it:

```
install.packages("tidycls/plot")
```

```
library(tidycls/plot)
```

Finally, there is an evolving package, named tidycls/analysis. (that wraps datasets in a convenient format, rendering them immediately usable from tidycls. Let's get this as well, as we're going to use one of the datasets in provide.

```
remotes::install_github("tidycls/tidycls/analysis")
```

```
library(tidycls/analysis)
```

Get the dataset

"Guess the correlation" is a fun dataset that tasks one—a person, if they feel like, or a program, if one is interested—to detect Kendall correlation between two variables displayed in a scatterplot.

tidycls/analysis will download, prepare, and preprocess it for us.

The training set is huge: it has 100000 observations. For exploratory purposes, we don't really need so much data—we'll restrict ourselves to small subsets, for each of training, validation, and test sets.

```
train_nrows <- 1L:1000
validation_nrows <- 1001L:1200
test_nrows <- 1201L:1400
```

Now, the following snippet does the following:

- download and cache the dataset,
- do some custom preprocessing on the images (on top of what is already done by default)—more on that soon
- take just the first 1000 observations and put them in train_ds (tidycls.Dataset object named train_ds,

```
train_ds <- gnn_cls_ds(.slicing_fn = function(ds){
  ds[train_nrows,] %>% .$.filtered_tbl %>% .$.data %>%
  .[,c("id", "image", "corr")]
  ,
  .load = TRUE,
  .preprocessing_fn = function(ds){
    ds %>%
    .$.filtered_tbl %>%
    .$.data %>%
    .[,c("id", "image", "corr")] %>%
    .$.data %>%
    .[,c("image")] %>%
    .$.data %>%
    .[,c("image")] %>%
    .[,1:3,,] %>%
    .[,,,c(2,1,3)] %>%
    .[,,,c(1,2,3)] %>%
    .[,,,c(1,2,3)] %>%
    .[,,,c(1,2
```