

“Bigger Data
Beats Better
Math, No
Question”

“If I invest in more
data, then the same
model will perform
better than switching
to a different ML
architecture ”

Bigger Data v Better Math

What has more impact in machine learning?

Brent Schneeman
@schnee

The Data



0
1
2
3
4
5
6
7
8
9 9

Fashion MNIST

<https://github.com/zalandoresearch/fashion-mnist>

70k split 60k training, 10k testing, with these labels

Designed to be a drop in replacement for Digits MNIST

Designed to be more difficult than Digits

Each observation consists of a label and 784 numbers {0,255}, which is a 28x28 grey scale image

Label	Meaning
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle Boot



Replacing Digits with Fashion? Easy

Change this line:

```
mnist <- keras::dataset_mnist()
```

Into this one

```
mnist <- keras::dataset_fashion_mnist()
```

Operations on either **mnist** object use exact same API

And it looks like

```
> str(mnist)
List of 2
$ train:List of 2
..$ x: int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
..$ y: int [1:60000(1d)] 9 0 0 3 0 2 7 2 5 5 ...
$ test :List of 2
..$ x: int [1:10000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
..$ y: int [1:10000(1d)] 9 2 1 1 6 1 4 6 5 7 ...
```

The Math

(these are not
great models, and
not greatly tuned)

RandomForest

```
num_trees <- 25
dim(x_train) <- c(nrow(x_train), 784)
dim(x_test) <- c(nrow(x_test), 784)
x_train <- x_train / 255
x_test <- x_test / 255

rf <- randomForest(x_train, as.factor(y_train),
                    ntree=num_trees)
```

xgboost

```
dim(x_train) <- c(nrow(x_train), 784)
dim(x_test) <- c(nrow(x_test), 784)

x_train <- x_train / 255
x_test <- x_test / 255

dtrain <- xgb.DMatrix(x_train, label = y_train)
dtest <- xgb.DMatrix(x_test, label = y_test)
train.gdbt<-xgb.train(params=list(objective="multi:softprob",
                                      num_class=10, eval_metric="mlogloss",
                                      eta=0.2, max_depth=5,
                                      subsample=1, colsample_bytree=0.5),
                        data=dtrain,
                        nrounds=150)
```

“Deep” Neural Net

```
model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(784)) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 10, activation = 'softmax')
```

Convolutional Neural Net

```
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu',
                input_shape = c(28, 28, 1)) %>%
  layer_conv_2d(filters = 64,
                kernel_size = c(3,3), activation = 'relu') %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_flatten() %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = num_classes, activation = 'softmax')
```

So we have
The Data
and we have
The Math



LET THEM FIGHT.

The Test Bench

```
tib <- c(1:9 / 100, 1:9 / 10, 91:100 / 100) %>%
  sort() %>%
  map_dfr(run_size_exp, x_train, y_train, x_test, y_test)
```

Basically, train with 1%, 2%, 3%, ..., 10%, 20%, 30%, ..., 98%, 99%, 100% of the training data and infer against the test set and you get

```
frac,exp_name,acc,auc
0.01,rf,0.8291,0.9759206200228384
0.01,xgb,0.8513,0.9859652247914469
0.01,dnn,0.8491,0.9860249426097214
0.01,cnn,0.9018,0.9922435869596523
```

run_size_exp

```
samples <- sample(nrow(x_train), floor(nrow(x_train) * frac), replace=FALSE) - 1  
  
x_t <- x_train[samples,,]  
y_t <- y_train[samples]  
  
data <- list(x_train = x_t,  
              y_train = y_t,  
              x_test = x_test,  
              y_test = y_test)  
  
math <- c(rf_exp, xgb_exp, dnn_exp, cnn_exp)  
  
preds <- math %>%  
  map(exec, !!!data)
```

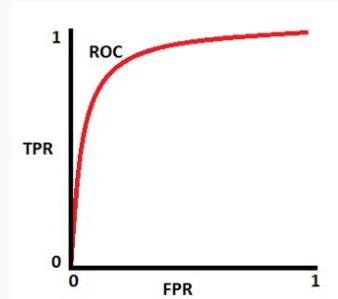
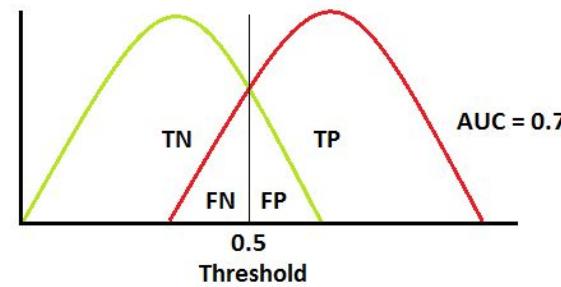


ML Metrics: Accuracy, AUC

Accuracy: how well the model predicted the true class, compared to predictions of all other classes.

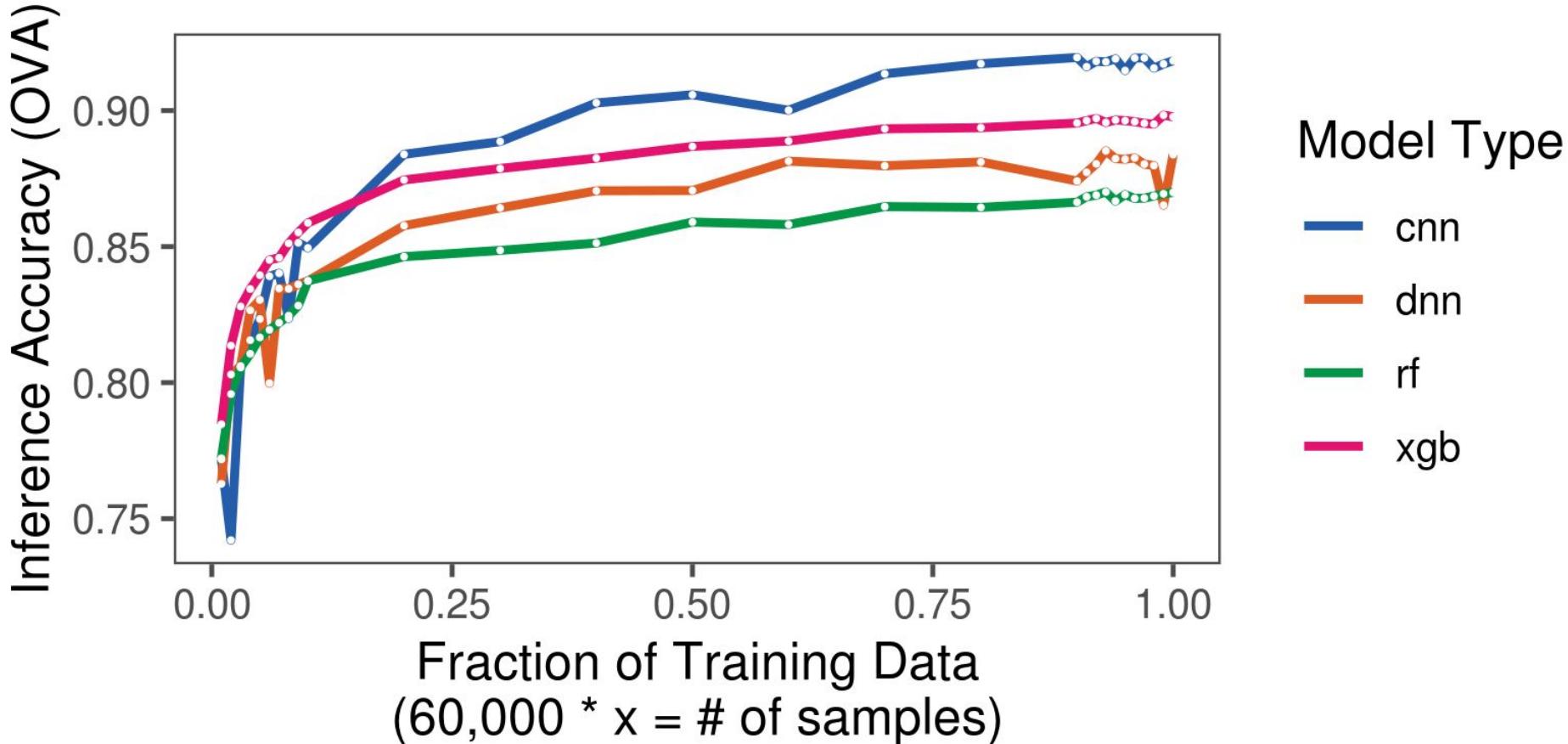
AUC: Area Under the Curve:

How well the model is able to separate classes



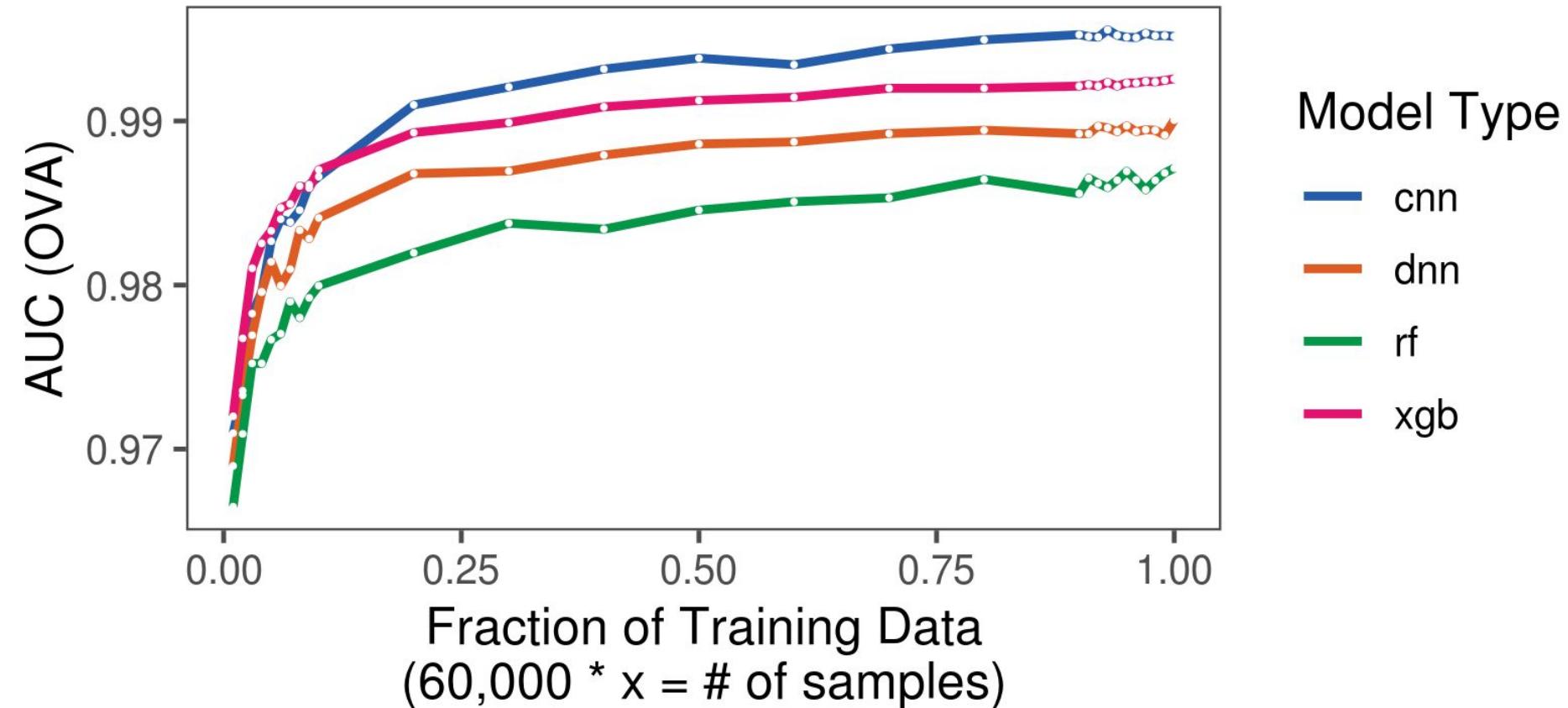
Model Architectures and Training Batch Size

Fashion MNIST Dataset



Model Architectures and Training Batch Size

Fashion MNIST Dataset

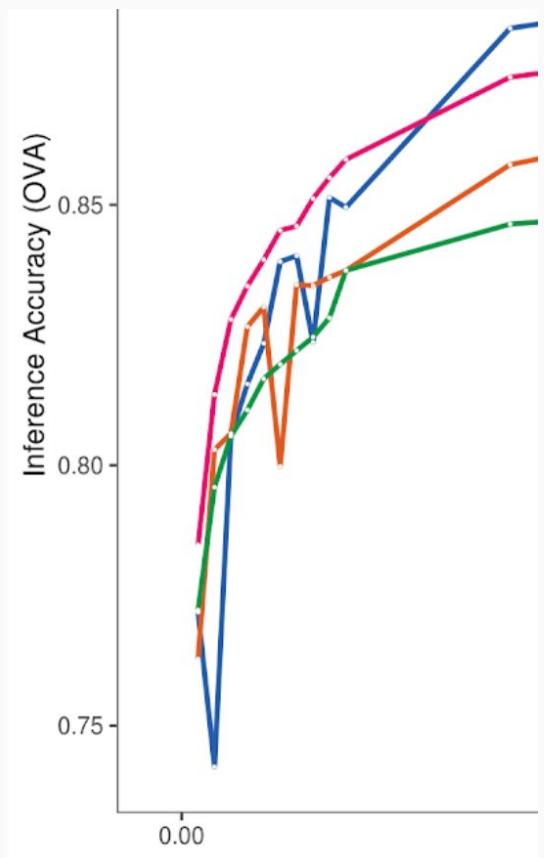
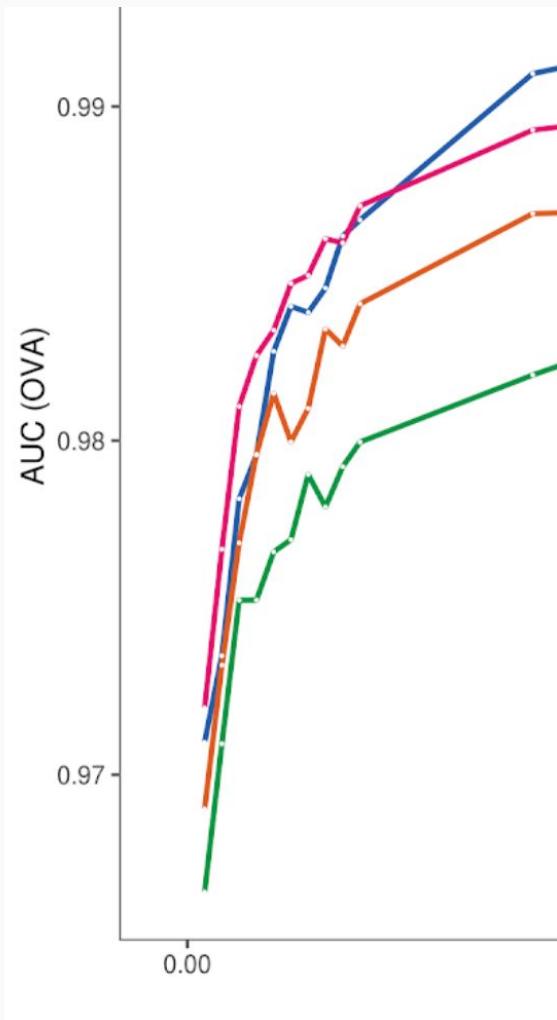


What did we
learn?

LHS of the Data

At 1%, we have 60 observations per class. 2% is 120 observations...

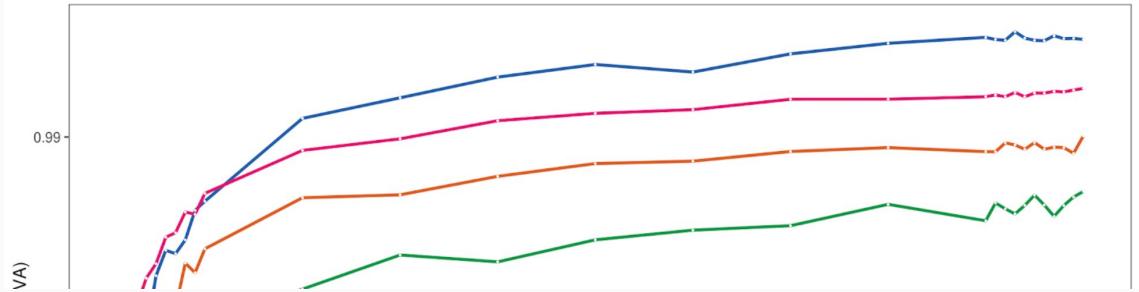
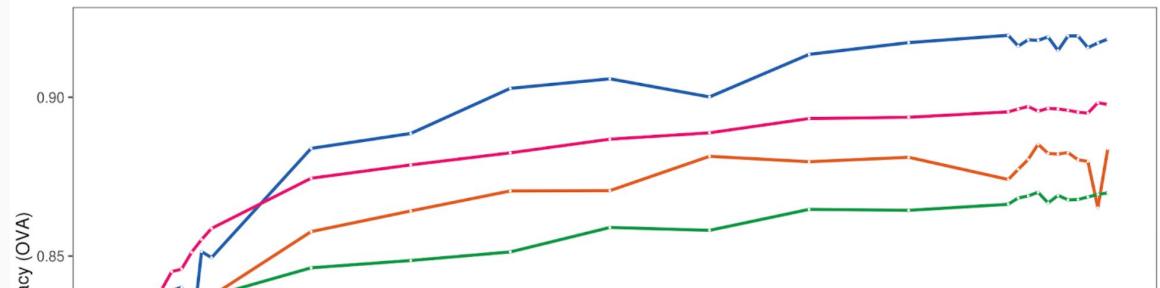
- Adding training data leads to steep improvement in response
 - BIGGER DATA!
- Moving to different models leads to steep improvements
 - BETTER MATH!



RHS of the Data

At 30%, we have 1800 observations per class. 40% is 2400 observations...

- Adding training data leads to moderate improvement in response
 - BIGGER DATA!
- Moving to different models leads to step improvements
 - BETTER MATH!



“Bigger Data
Beats Better
Math, No
Question”

WHOOPS!

If you are already
at the “BEST”
MATH, BIGGER
DATA is a good
strategy

In practice: use whatever works best
on ImageNet...

Andrej Karpathy, CS231N

What about
Better Data?



What is Better Data?

Training a classifier (“shirt, trouser, handbag, …”) requires training data.

Training a good classifier requires high-confidence in your training data.

Fashion has 60,000 observations for training: 6,000 examples of each class
that we assume are correct to learn from

What if that assumption is not correct?

How can we invalidate the assumption of correctness?

That's right,
expose the data
labels to Pure Evil

Teratogenic Effects of Pure Evil in Ursus Teddius Domesticus

Dr. Allison von Lonsdale
Institute for Dangerous Research
Department of Mad Biology

Protocols

1. A sample of Pure Evil was obtained from the ruins of an exploded toaster in the south of England.
2. Pure Evil was administered, via drinking water, to pregnant laboratory teddy bears for the duration of their pregnancy (4 months).
3. Dosage varied from 0 parts per million (ppm) to 1000ppm, titrating upwards by steps of 100ppm.
4. Offspring were euthanized and mounted for display.

DON'T TOUCH IT
IT'S EVIL



Observations

1. Pure Evil has strong teratogenic effects, with a tendency to produce asymmetry.
2. These effects vary widely, without a clear progression along a phenotypic path.
3. At higher doses, there can be inclusions of tissue from other species (kingdom? phyla?) we have yet to identify that purple wiggly stuff in specimen #7.

Notes

1. Specimen #9: This lump of discolored stuffing was miscarried at 2 months.
2. At the dosage of 1000ppm, pregnancy appeared to be reabsorbed before the 2 month mark. Administration of Pure Evil at the same dosage continued. By the 3 month mark, subject developed dental hypoplasia, ocular hypoesthesia, and extreme behavioral changes. At 3 months 2 weeks, subject chewed through the steel bars of its cage, after which it killed and partially consumed two graduate students. Subject was then euthanized with a sustained burst of automatic weapons fire.

Conclusions

Pure Evil's teratogenic effects indicate that it should be treated at even low dosages as a hazardous material and treated as an extremely dangerous threat. Every effort should be made to prevent its entering the water table and its food chain. While it is true the potential risks of Pure Evil are low, the potential risks of higher doses to be used in the development of weaponized life-forms, the difficulty of safely disposing of their bodies, and the risks of environmental contamination in the event of escape, strongly skew the cost-benefit analysis in the direction of *Holy Crap Don't Use This Stuff*.



That's the evilest thing I can imagine.

Applying evil to training data

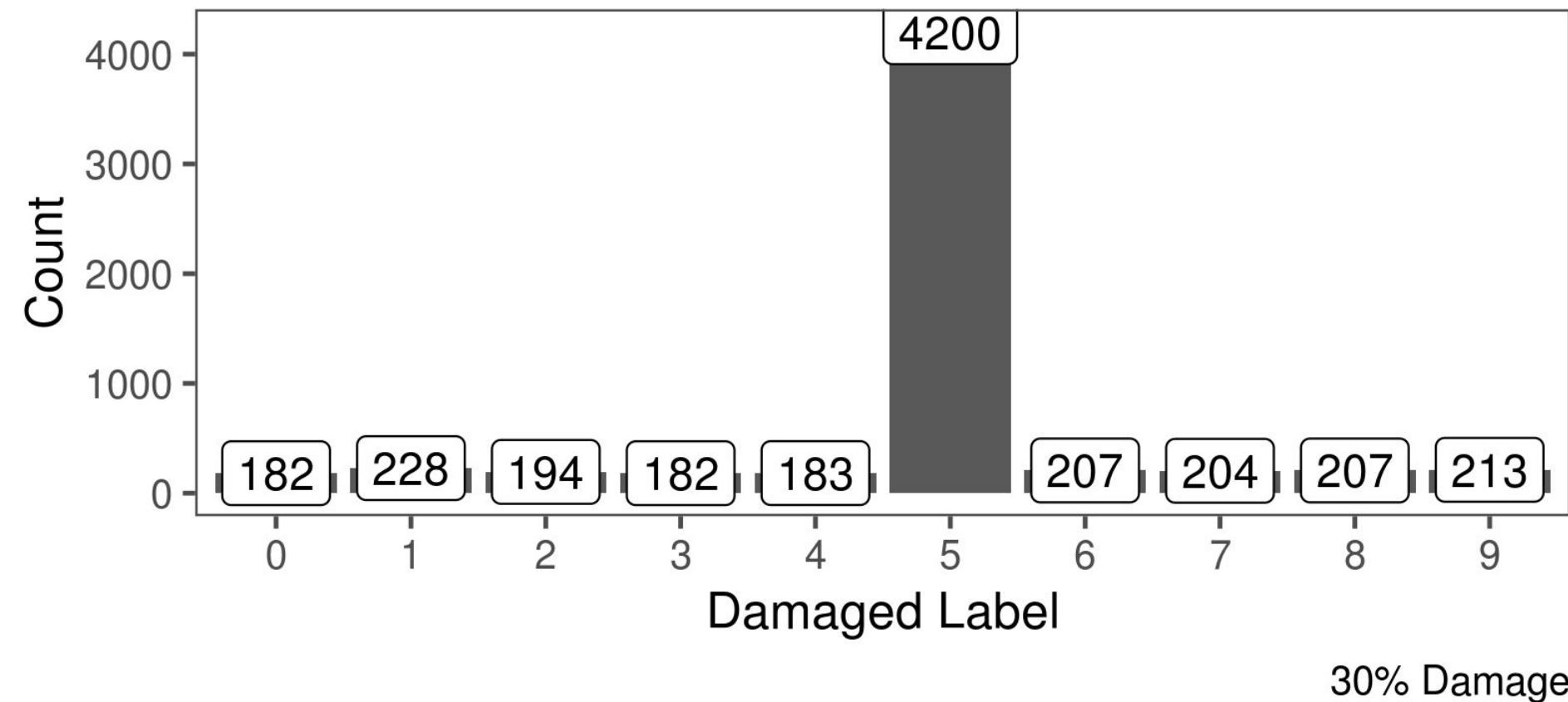
Damage the labels

```
damage_tib <- c(0:9 / 100, 1:9 / 10, 91:99 / 100) %>%
  sort() %>%
  map_dfr(run_random_damage_exp, x_train, y_train, x_test, y_test)
```

Randomly changes a “true” label to one of the other 9, and then train and test

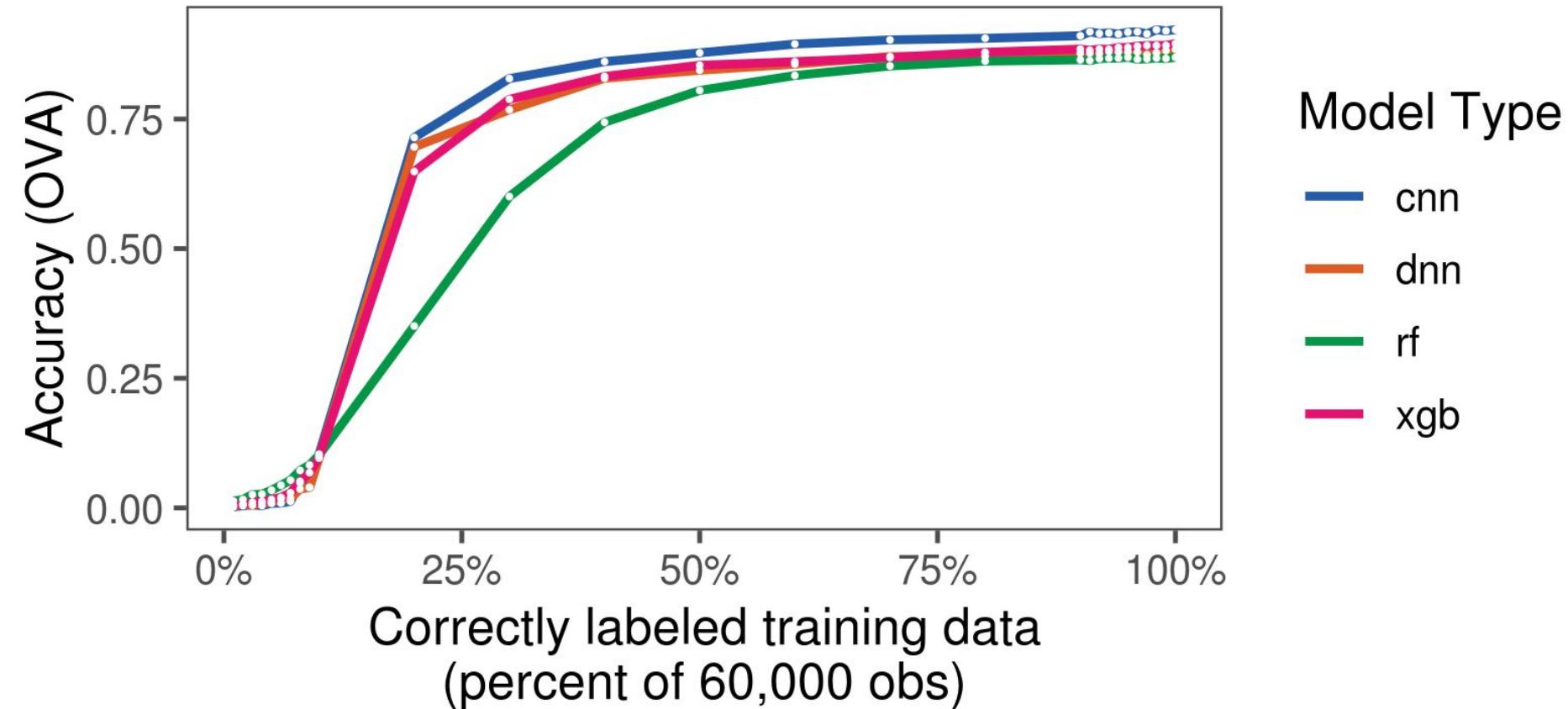
Randomly Biased Labels

True Label = 5



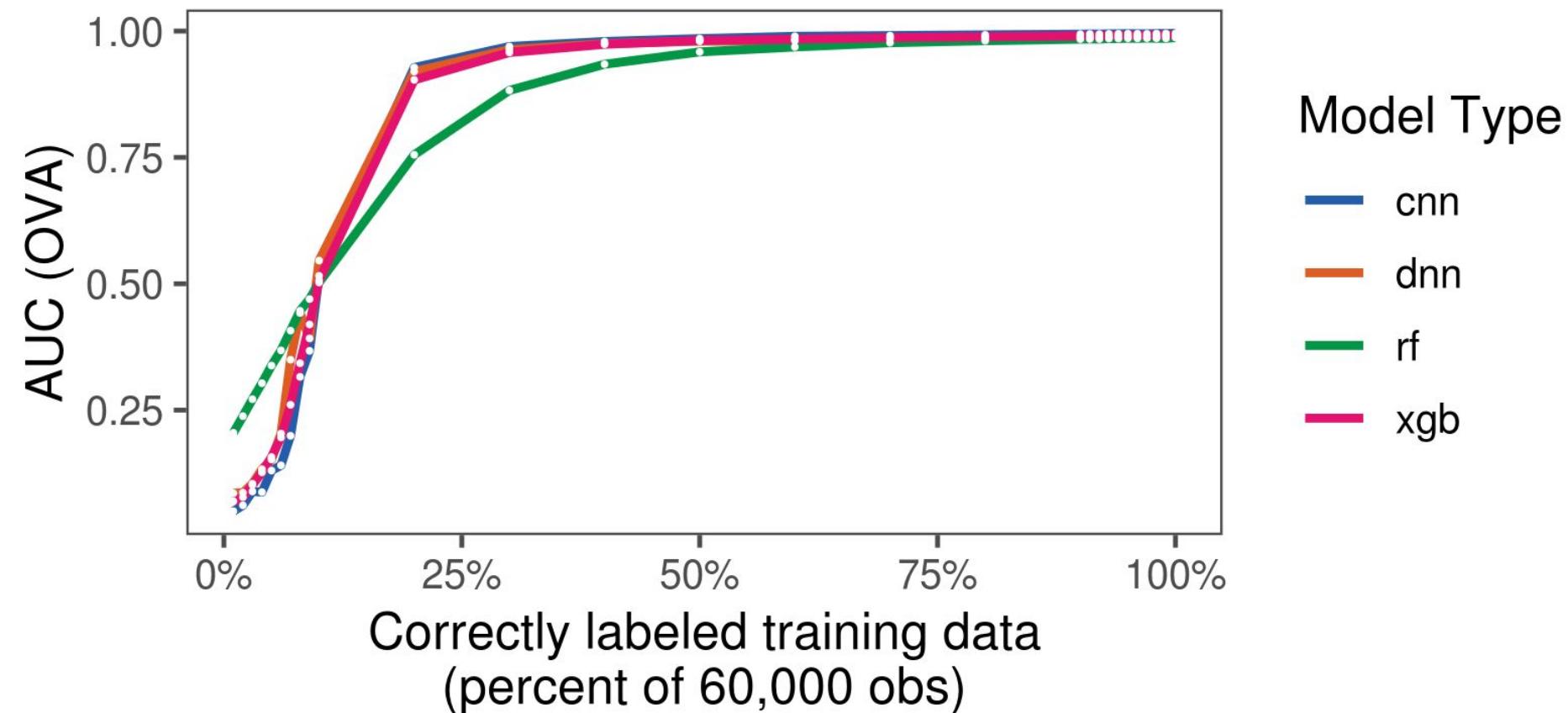
Model Architectures and Random Bias

Fashion MNIST Dataset



Model Architectures and Random Bias

Fashion MNIST Dataset



But is that really
the most evil we
can be to data?

Random Bias -> Constant Bias

With a random bias, every mis-label has a 1-in-9 chance of being some other label - the counter-signal is smeared out

Machine Learning needs strong signals to learn and the information content in the 1-in-9 isn't relatively strong

Will a constant bias be more damaging?



Applying evil to training data, take 2

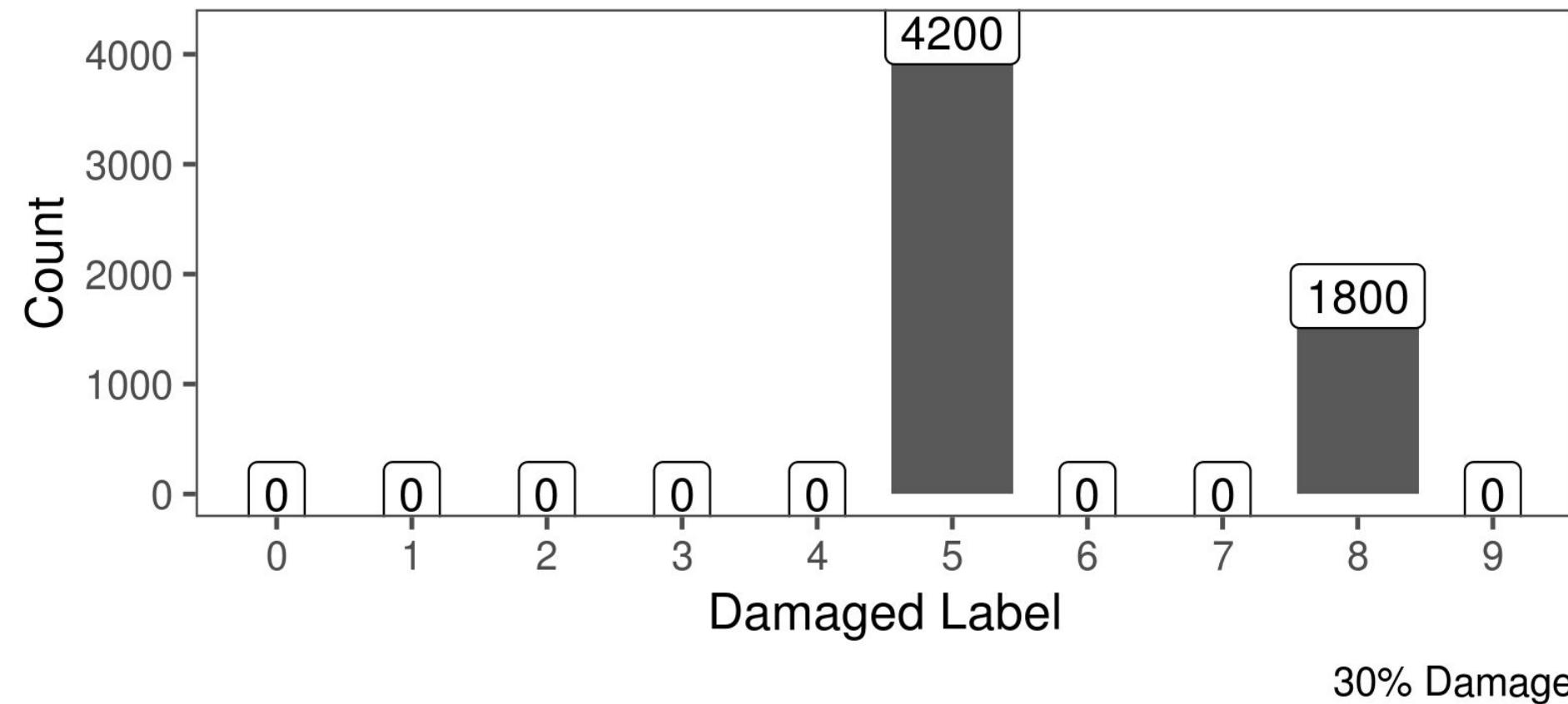
Damage the labels

```
damage_tib <- c(0:9 / 100, 1:9 / 10, 91:99 / 100) %>%
  sort() %>%
  map_dfr(run_constant_damage_exp, x_train, y_train, x_test, y_test)
```

Changes a label consistently to another label (“0 becomes 1, always”), and then train and test

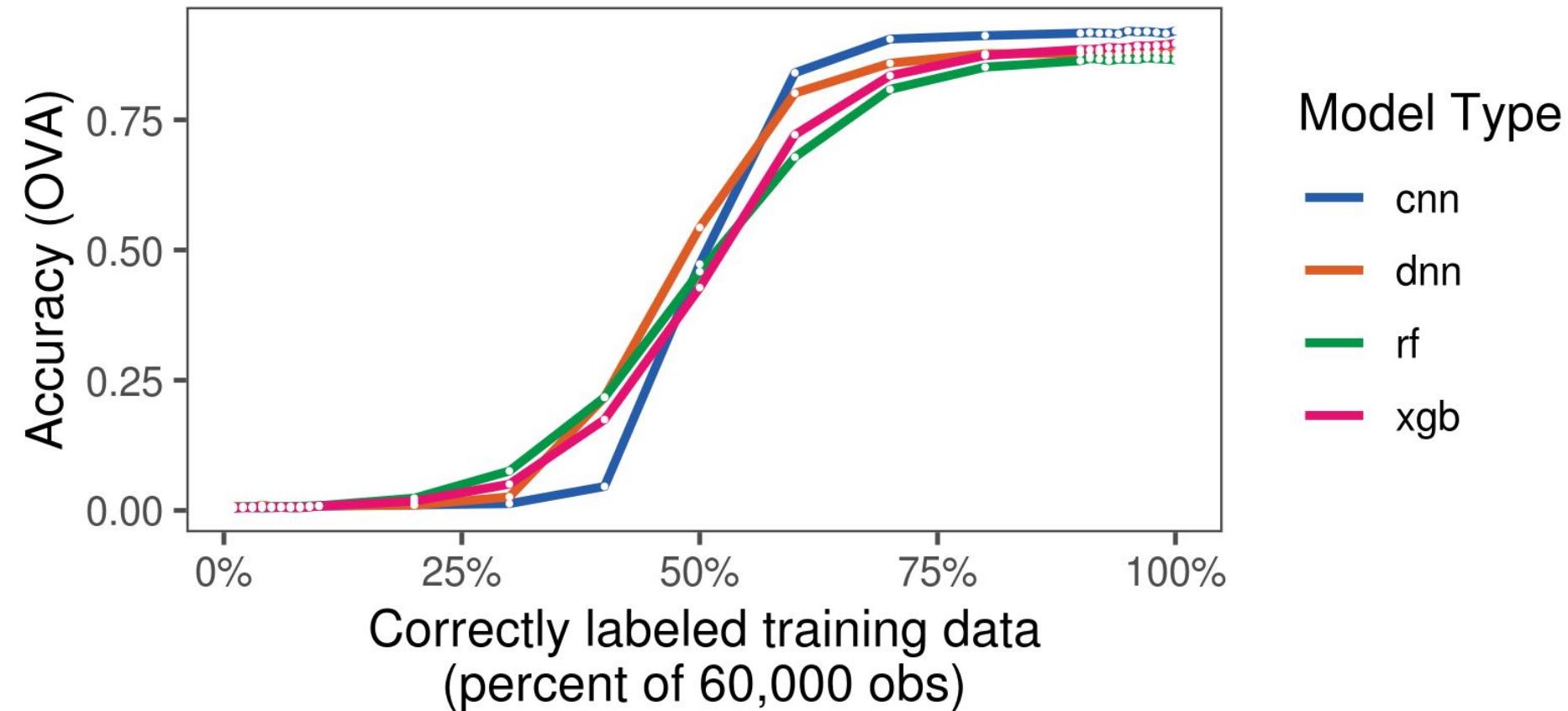
Constant Biased Labels

True Label = 5



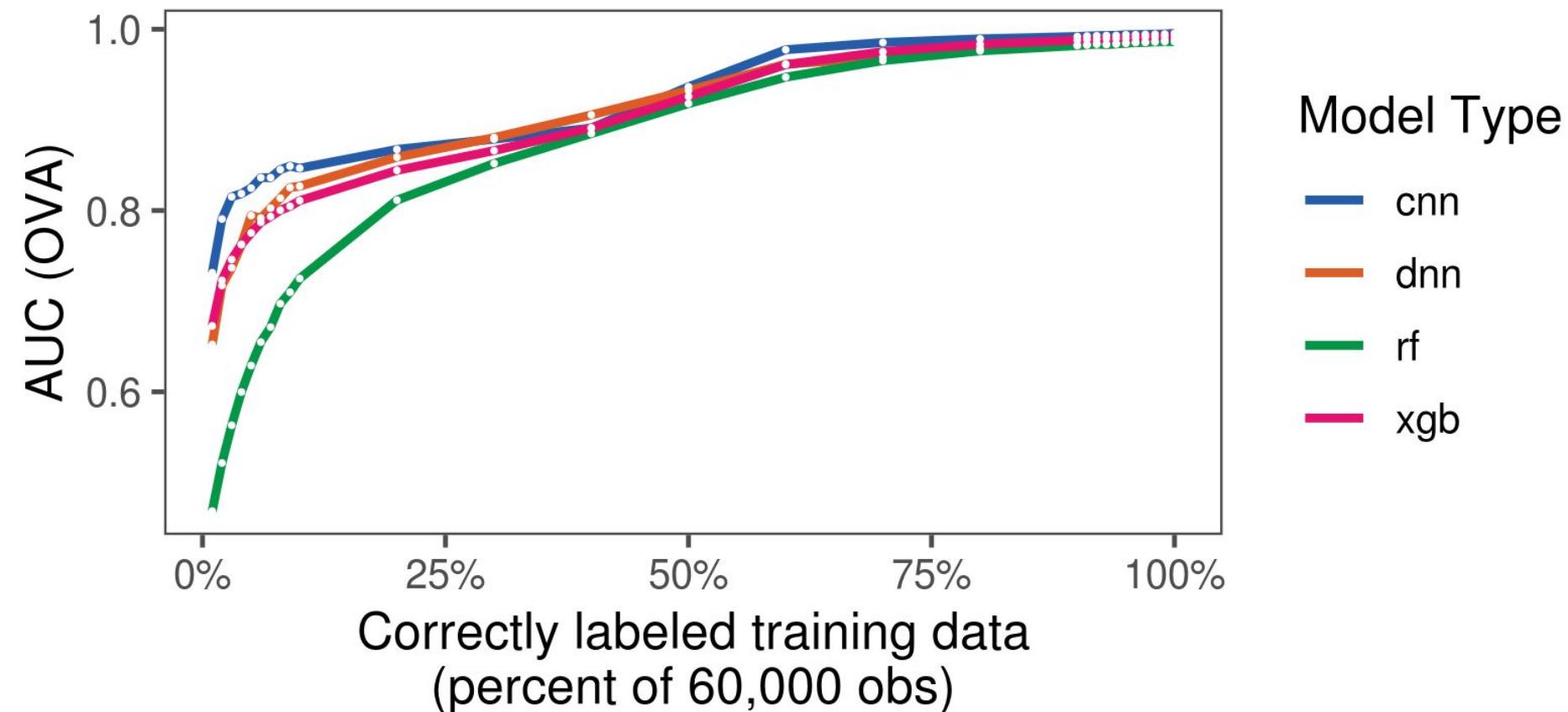
Model Architectures and Constant Bias

Fashion MNIST Dataset



Model Architectures and Constant Bias

Fashion MNIST Dataset



But is that *really*
the most evil we
can be to data?



Expose the data to a pack of middle school girls - there are some things I will not do to data

What if we ‘fix’
the architecture
and vary samples
and bias?

Some test bench code

```
damage_frac <- c(0:5 / 10) %>%
  sort()

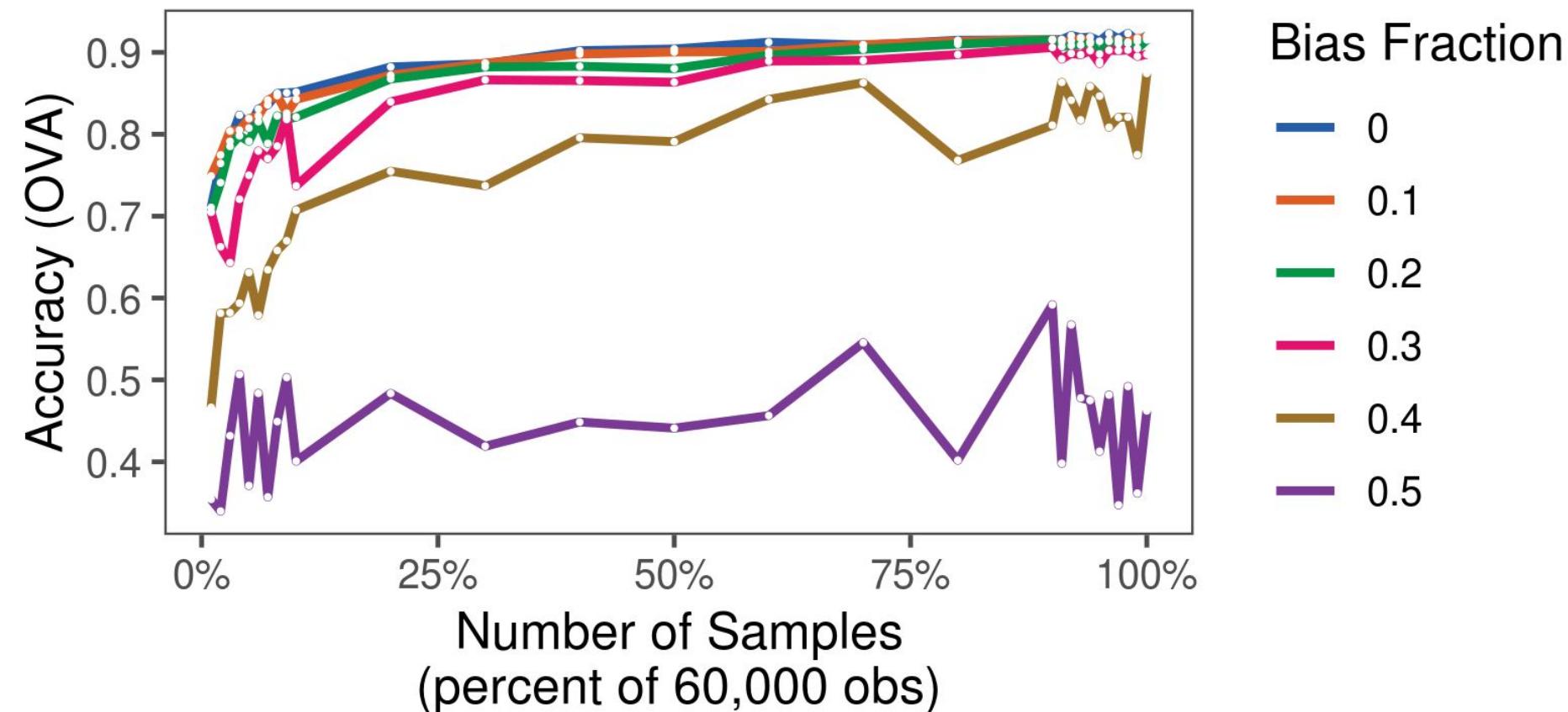
sample_frac <- c(1:9/100, 1:9 / 10, 91:100 / 100) %>%
  sort()

cross_prod <- cross_df(list(d=damage_frac, s=sample_frac))

results_tib <-
  map2_dfr(cross_prod$d, cross_prod$s, run_cnn_damage_exp,
  x_train, y_train, x_test, y_test)
```

CNN Architecture and Bias

Fashion MNIST Dataset

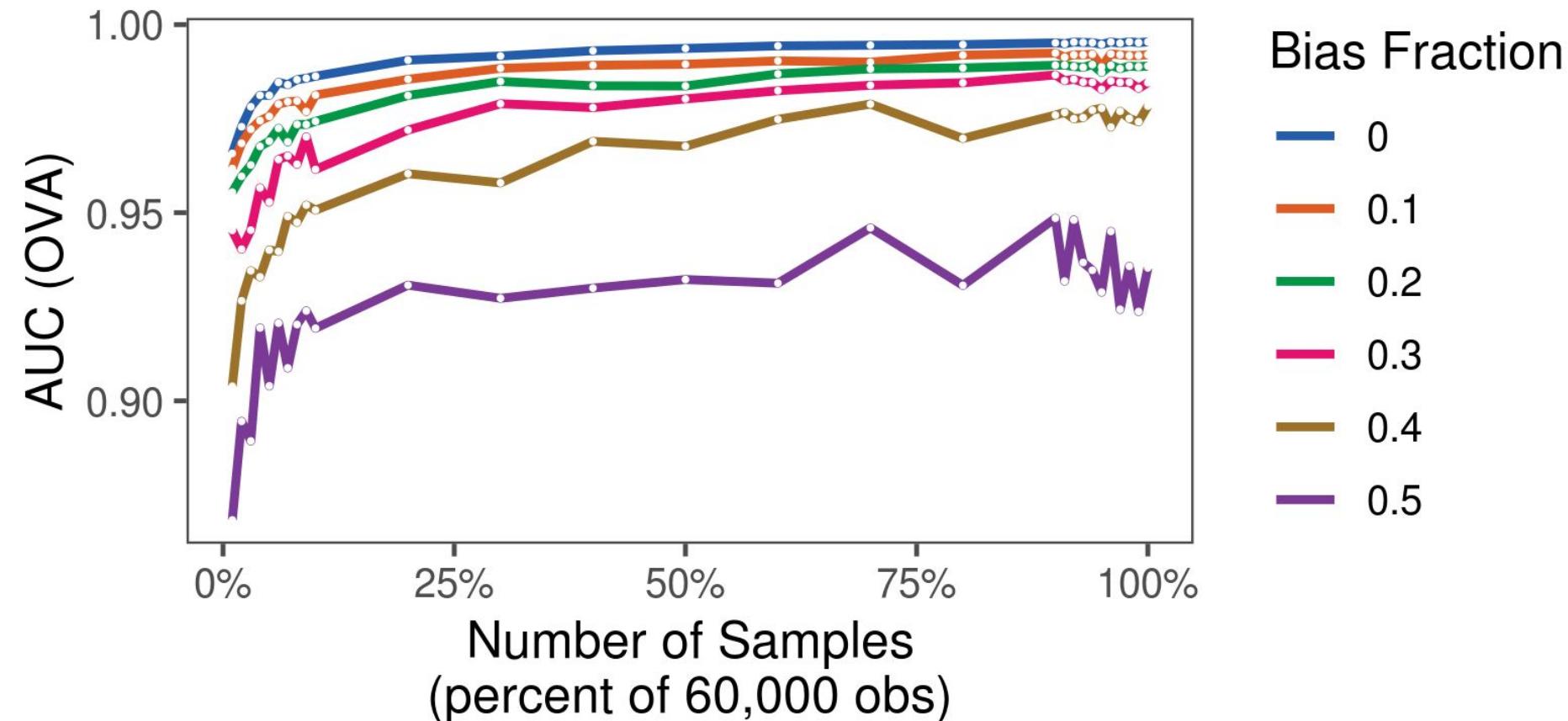


Observations needed for > 90% accuracy

quality	acc	frac_sample	num_obs	d_obs
1.0	0.902	0.40	24000	NA
0.9	0.900	0.50	30000	0.25
0.8	0.904	0.70	42000	0.40
0.7	0.902	0.94	56400	0.34

CNN Architecture and Bias

Fashion MNIST Dataset



Observations needed for > 98% AUC

quality	auc	frac_sample	num_obs	d_obs
1.0	0.981	0.05	3000	NA
0.9	0.981	0.10	6000	1.0
0.8	0.981	0.20	12000	1.0
0.7	0.980	0.50	30000	1.5

An economic argument for quality

$$\frac{\partial \text{data size}}{\partial \text{quality}} \ll -1$$

One step in quality does not equal a similarly sized step in the number of observations

If data is hard to acquire, a little effort on quality goes a long way

Some thoughts
from the Internet



Max Lin @m4x1n · May 14

Just like doubling **training data** won't automatically give you 2x improvement on **#MachineLearning** performance, think again when someone says more data are better and why, and see more examples in "The Empty Promise of Data Moats" a16z.com/2019/05/09/dat...



JD Long @CMastication · Apr 26

When people ask me about the future of data science...

John Bovay @JohnBovay

Paul Romer on EconTalk: "over time, I've become more and more convinced of the importance of that pure hard work of collecting data and summarizing them in a manageable way" econtalk.org/paul-romer-on-...

4

8

43



Emily @servemethesky · May 11

Amazon's recruiting tool was dismissing women candidates because the AI was trained with data that included more male faces than female faces. That **training data** was imperfect. It was an indirect path to discrimination.

#TEDxROC

1

1

3



In practice: use whatever works best on ImageNet. If you're feeling a bit of a fatigue in thinking about the architectural decisions, you'll be pleased to know that in 90% or more of applications you should not have to worry about these. I like to summarize this point as "*don't be a hero*": Instead of rolling your own architecture for a problem, you should look at whatever architecture currently works best on ImageNet, download a pretrained model and finetune it on your data. You should rarely ever have to train a ConvNet from scratch or design one from scratch.

Andrej Karpathy, CS231N



Alex Gude @alex_gude · Apr 24

Here is a real use case from work for model improvement and the steps taken to get there:

- Baseline: 53%
- Logistic: 58%
- Deep learning: 61%
- **Fixing your data: 77%**

Some good ol' fashion "understanding your data" is worth its weight in hyperparameter tuning!

21 354 1.3K



Chris Albon

@chrisalbon

Following
Supervised Deep Learning Rule Of Thumb
machinelearningflashcards.com

SUPERVISED DEEP LEARNING RULE OF THUMB

Number of observations per category of the target to get good performance:


Total Number of observations to match human performance:


BY CHRIS ALBON

12:16 PM - 5 Jun 2019

1. Become one with the data

The first step to training a neural net is to not touch any neural net code at all and instead begin by thoroughly inspecting your data. This step is critical. I like to spend copious amount of time (measured in units of hours) scanning through thousands of examples, understanding their distribution and looking for patterns. Luckily, your brain is pretty good at this.

Andrej
Karpathy's
ML Rubric

Closing Thoughts

Better Math

- Always worthwhile; research effort, unplannable at the limit

Bigger Data

- Always worthwhile, engineering effort usually

Better Data

- For a given ML metric, $\partial \text{data} / \partial \text{quality} \ll -1$. This implies that overcoming bias is costly

Thanks!

Brent Schneeman

schneeman@gmail.com

@schnee

<https://github.com/schnee/big-data-big-math>

(btw - I'm #OpenToWork)

