# Reusing trained models with TensorFlow

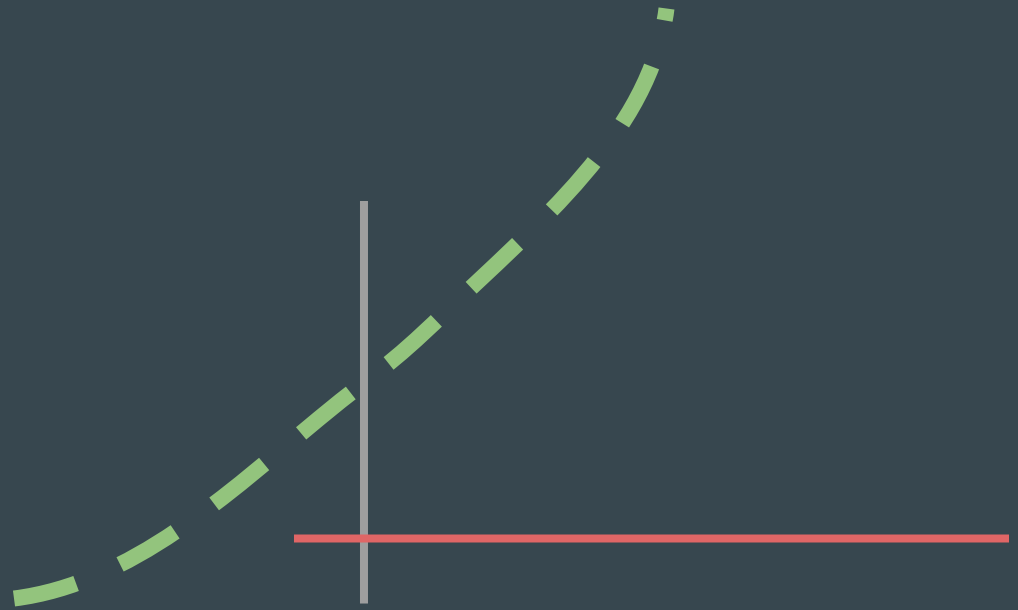
...

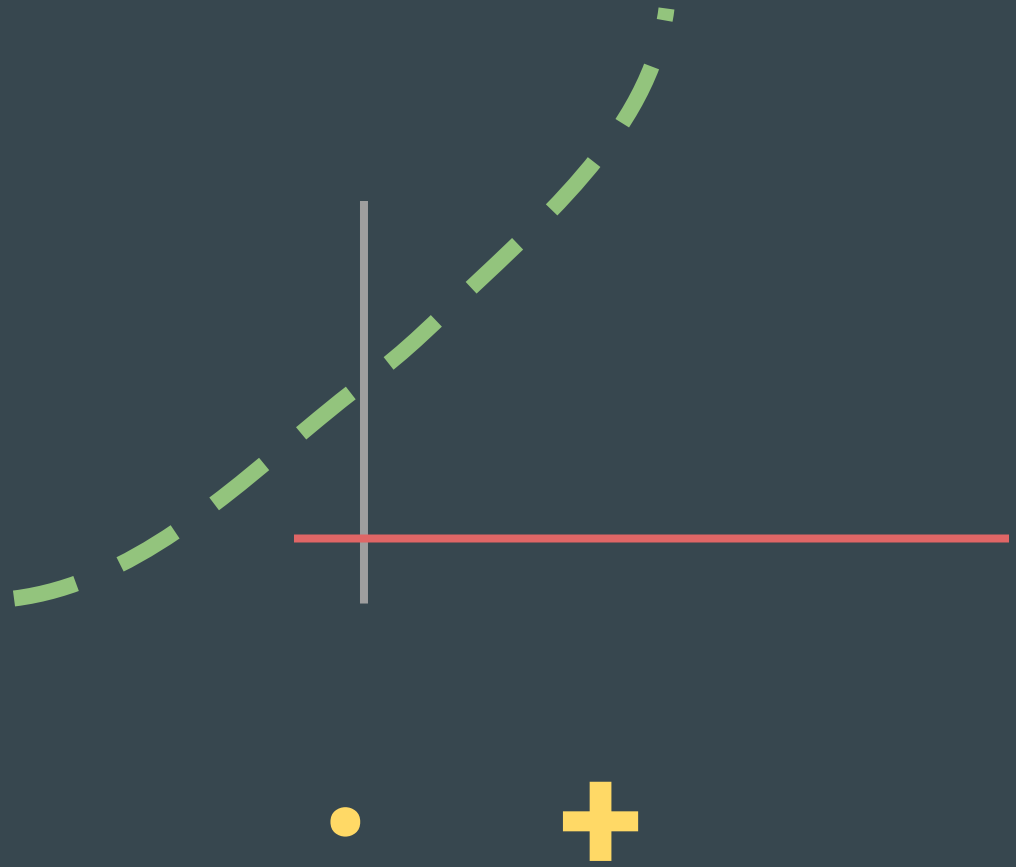
Alejandro Solano - PyCon PL 2017

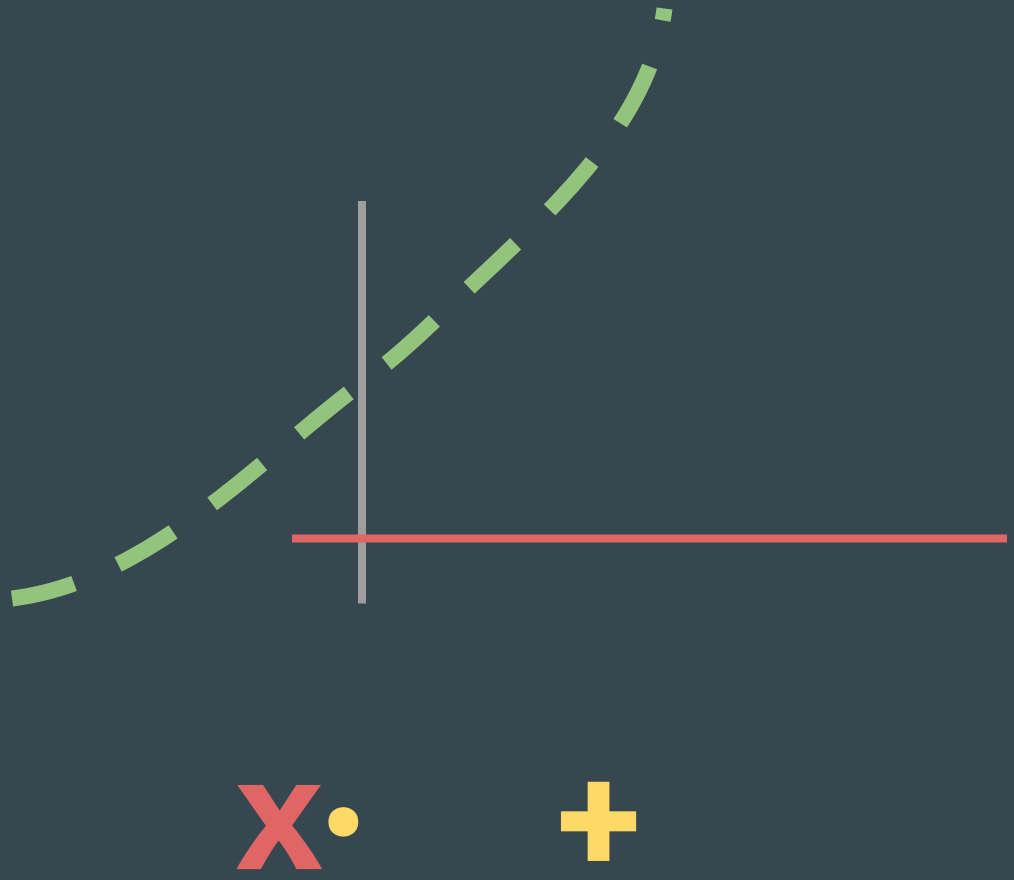
save

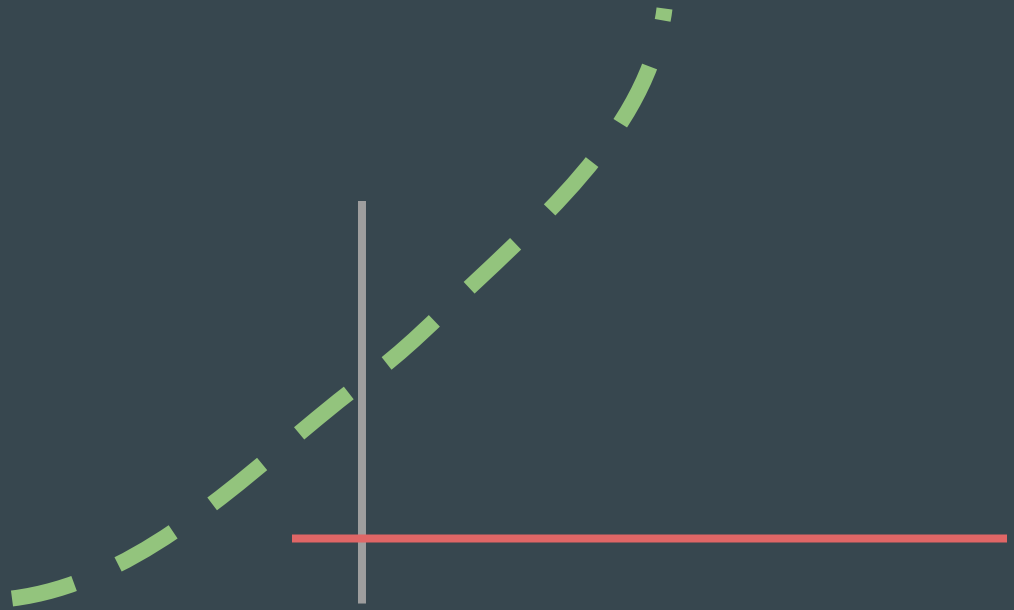
save load

save load run

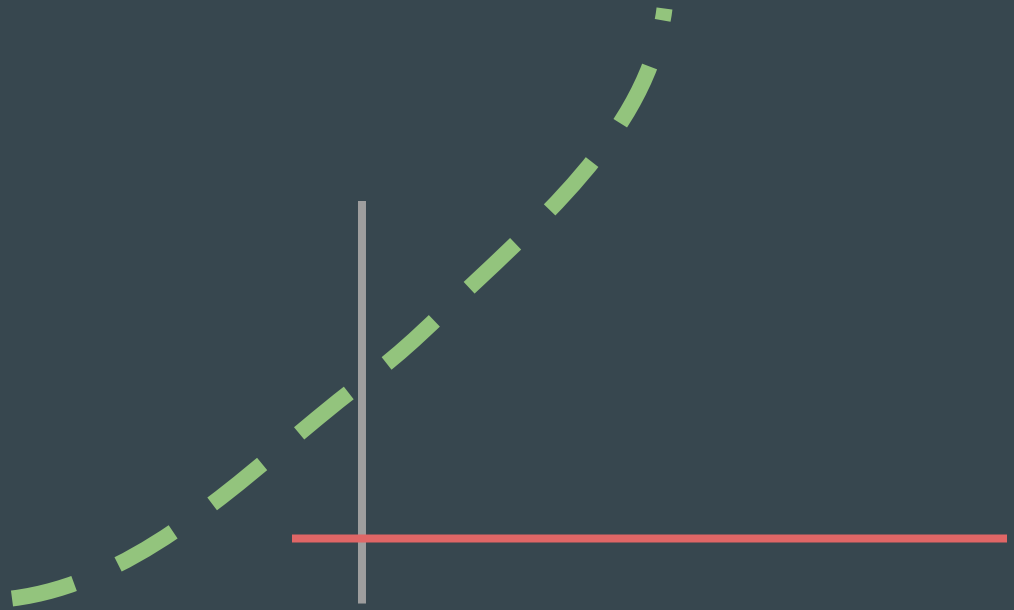




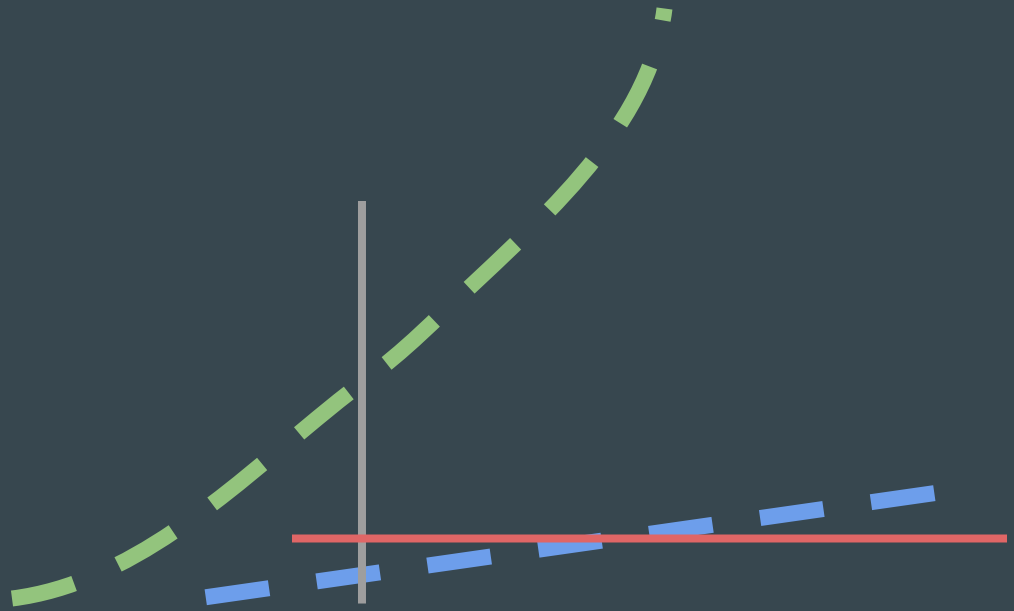




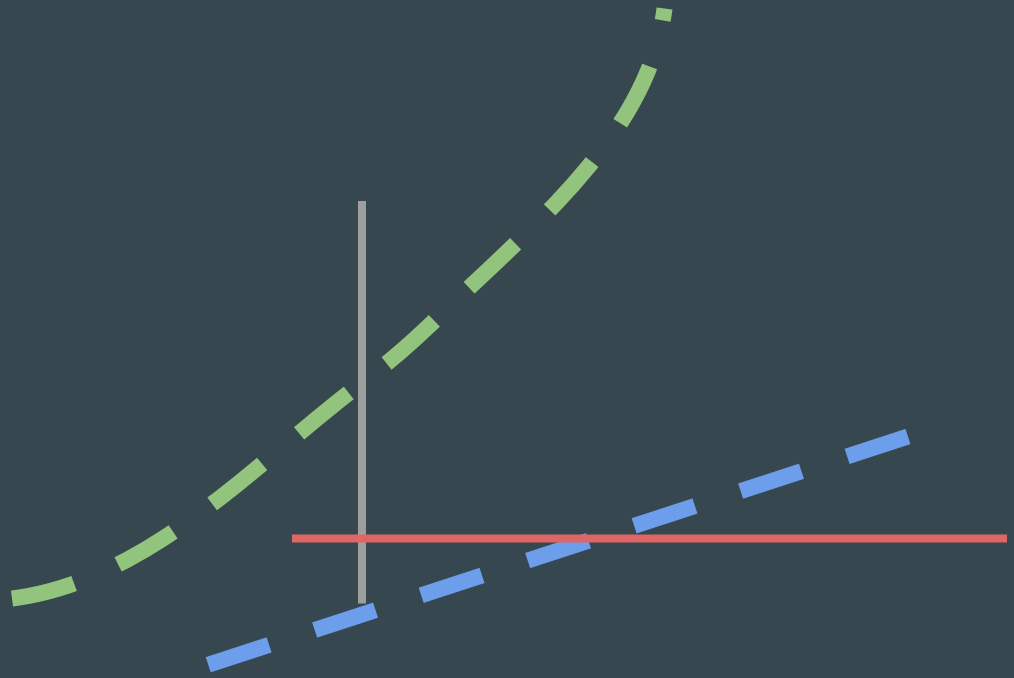
$$x \cdot W +$$



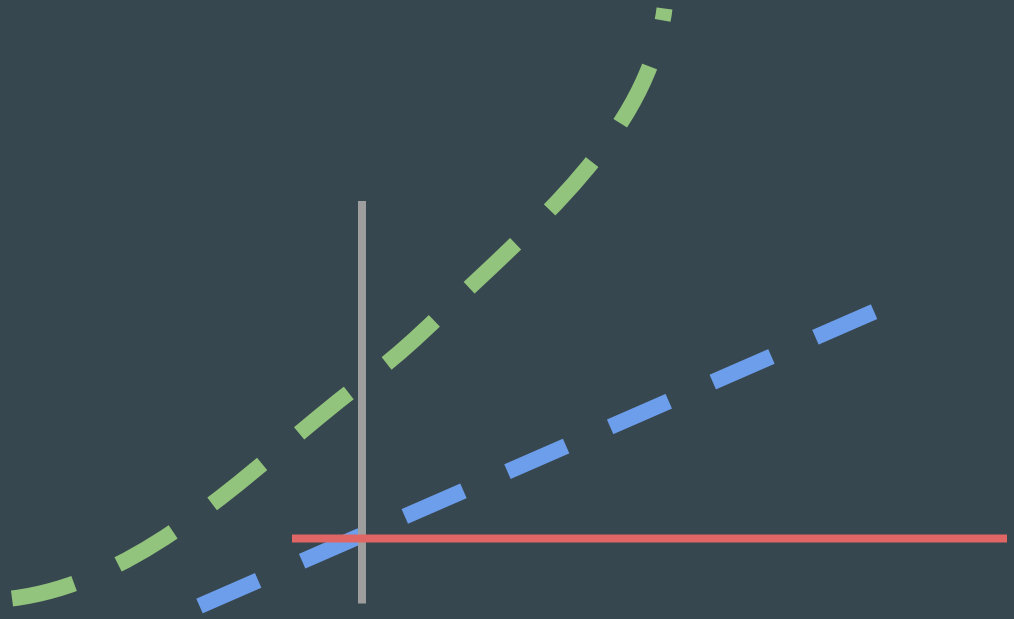
$$x \cdot W + b$$



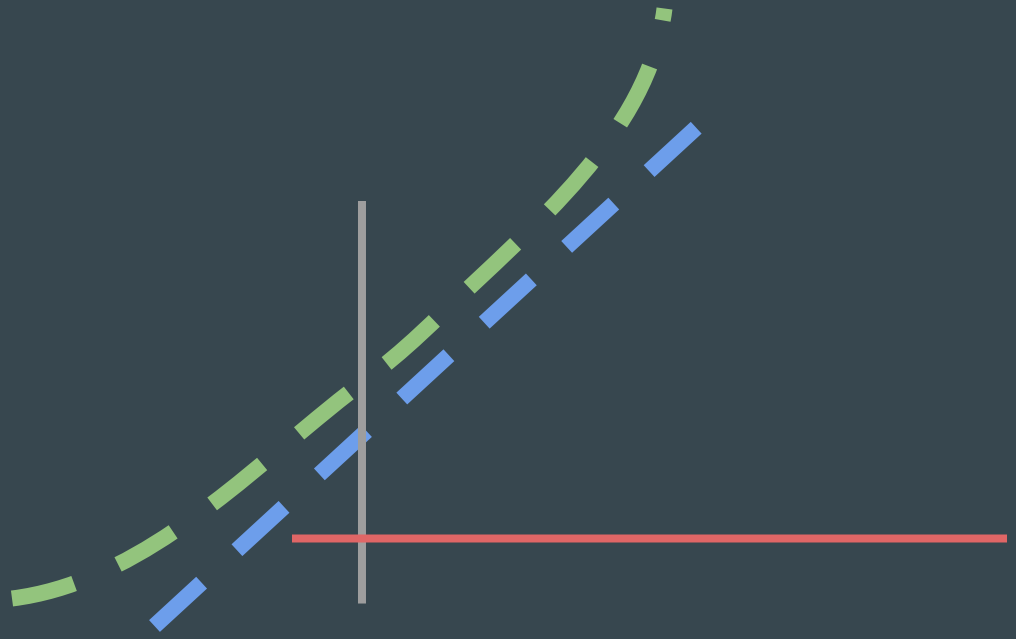
$$x \cdot 0.1 + -0.1$$



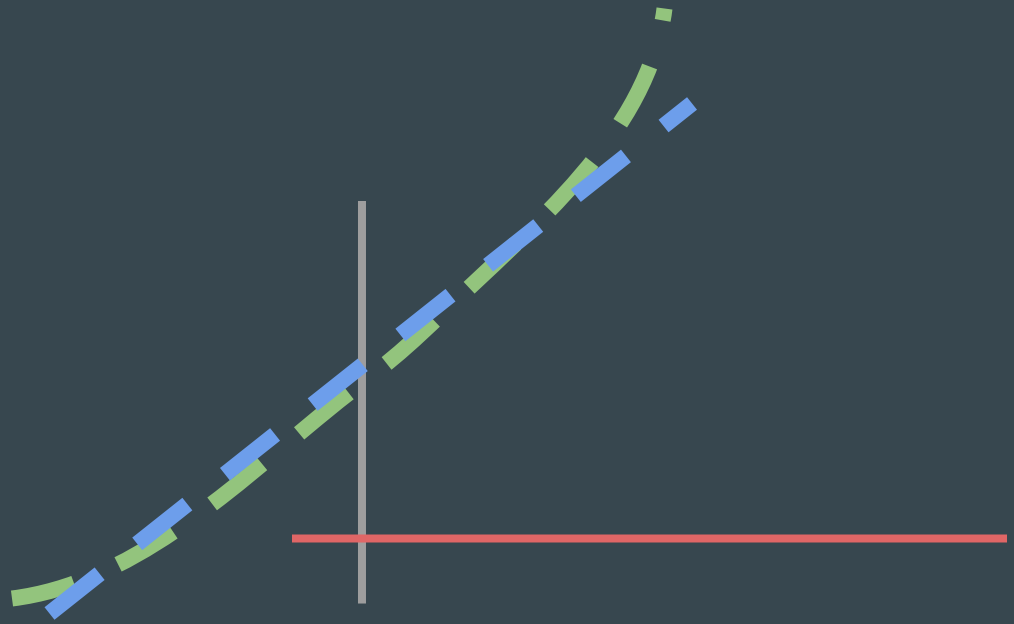
$$x \cdot 0.3 + -0.2$$



$$x \cdot 0.4 + 0.1$$

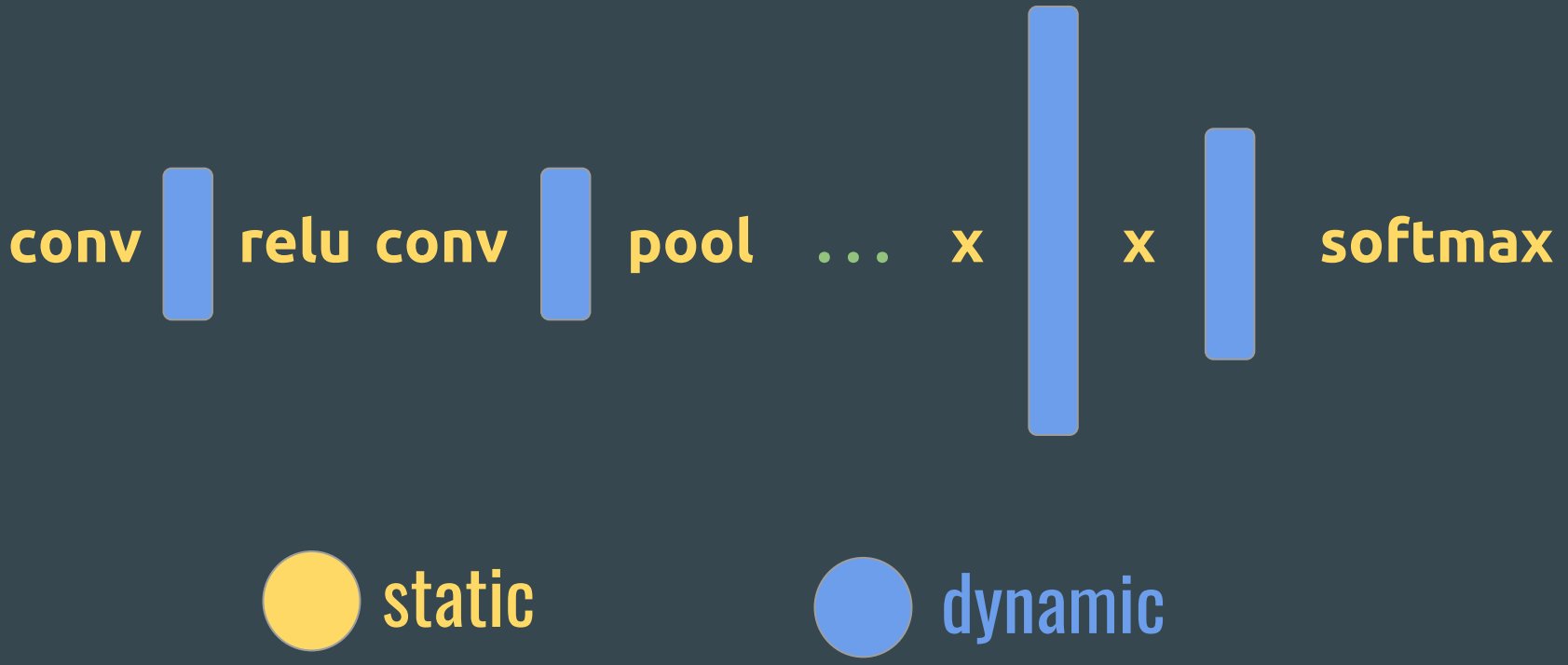


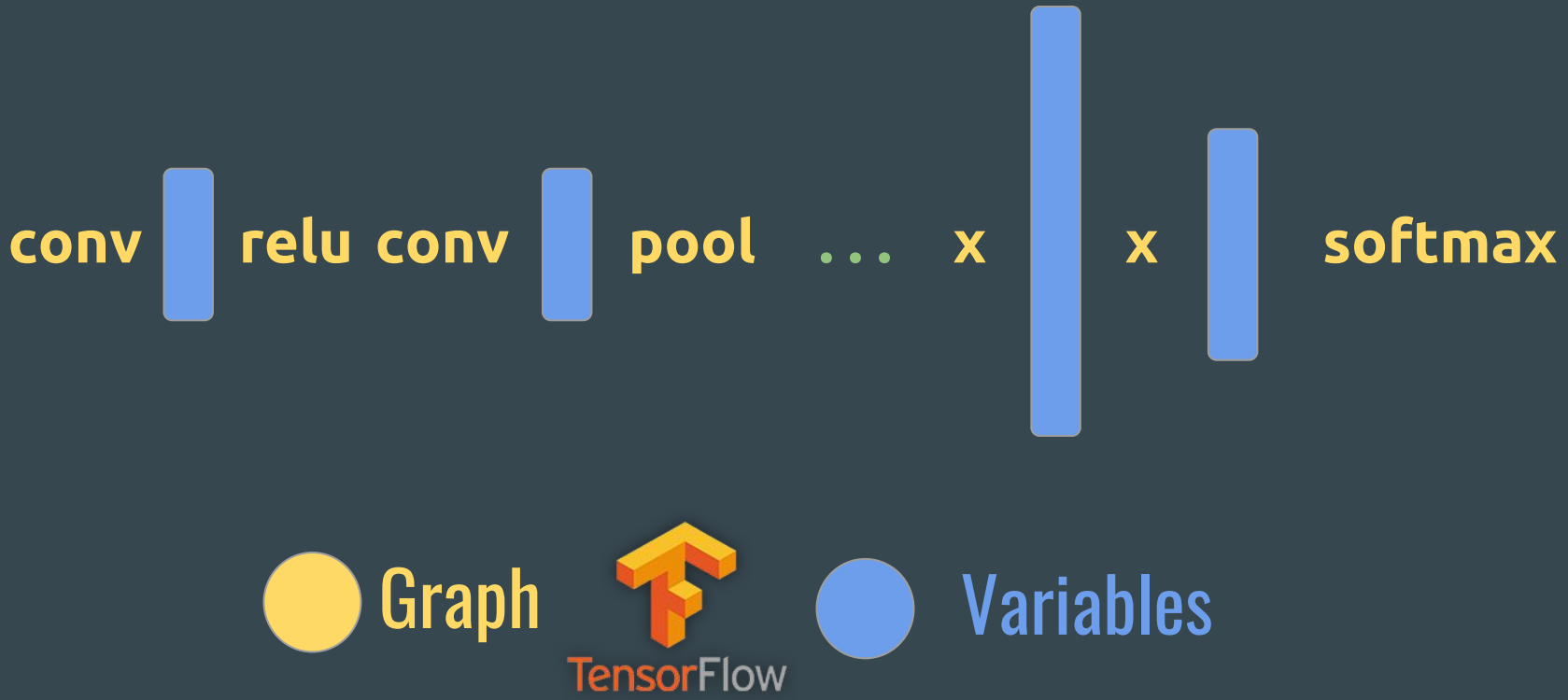
$$x \cdot 1.2 + 0.8$$



$$x \cdot 0.95 + 1.05$$


· 0.95 + 1.05
static
dynamic


conv
relu
conv
pool
. . .
x
x
softmax
static
dynamic


conv
relu
conv
pool
. . .
x
x
softmax
Graph
TensorFlow
Variables

# Checkpoints

# Checkpoints - Save

```
##### GRAPH #####
saver = tf.train.Saver()

##### SESSION #####
saver.save(sess, ‘model.ckpt’)
```

# Checkpoints - Save

```
##### GRAPH #####
saver = tf.train.Saver()

##### SESSION #####
saver.save(sess, ‘model.ckpt’)
```

.ckpt.meta .ckpt.index .ckpt.data

# Checkpoints - Save

```
##### GRAPH #####
saver = tf.train.Saver()

##### SESSION #####
saver.save(sess, ‘model.ckpt’)
```

.ckpt.meta

.ckpt

# Checkpoints - Load

.ckpt.meta

.ckpt

# Checkpoints - Load

.ckpt.meta

.ckpt

```
##### GRAPH #####
saver = tf.train.import_meta_graph('model.ckpt.meta')

##### SESSION #####
saver.restore(sess, 'model.ckpt')
```

# Run a loaded graph

```
##### SESSION #####
with tf.Session() as sess:
    preds = sess.run(output_tensor, feed_dict={
        input_placeholder: image
        })
```

# Run a loaded graph

```
##### SESSION #####
with tf.Session() as sess:
    preds = sess.run(output_tensor, feed_dict={
        input_placeholder: image
        })
```

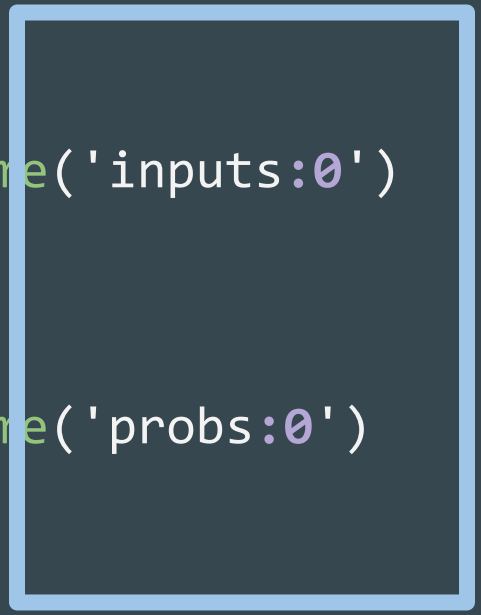
we need access to those tensors!

# Run a loaded graph

```
input_placeholder =
tf.get_default_graph().get_tensor_by_name('inputs:0')


output_tensor =
tf.get_default_graph().get_tensor_by_name('probs:0')
```

# Run a loaded graph

```
input_placeholder =
tf.get_default_graph().get_tensor_by_name('inputs:0')


output_tensor =
tf.get_default_graph().get_tensor_by_name('probs:0')
```



# Run a loaded graph

```
tensorboard
```

```
https://www.youtube.com/watch?v=eBbEDRsCmv4
```

# Checkpoints - Summary

| **Save** | Files | **Load** |
| --- | --- | --- |
| tf.train.Saver -> .save() | .ckpt.meta | saver = tf.train.import_meta_graph() |
| | .ckpt | saver.restore() |

# Checkpoints

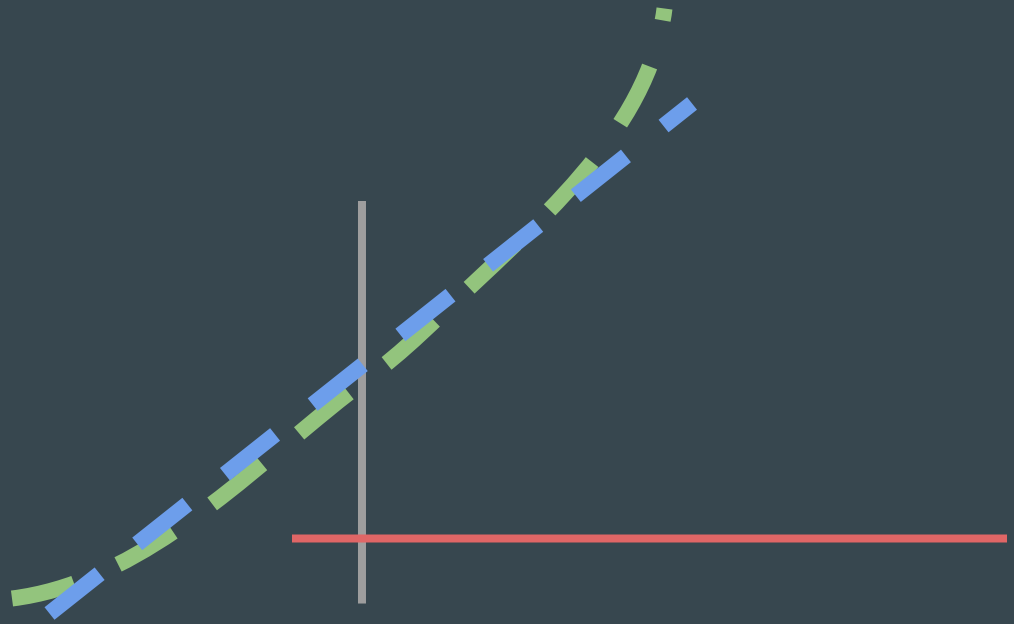
## Pros:

- Checkpoints keep the variables dynamic, so we can **retrain** the model after loading it.
- Easy to save and load.

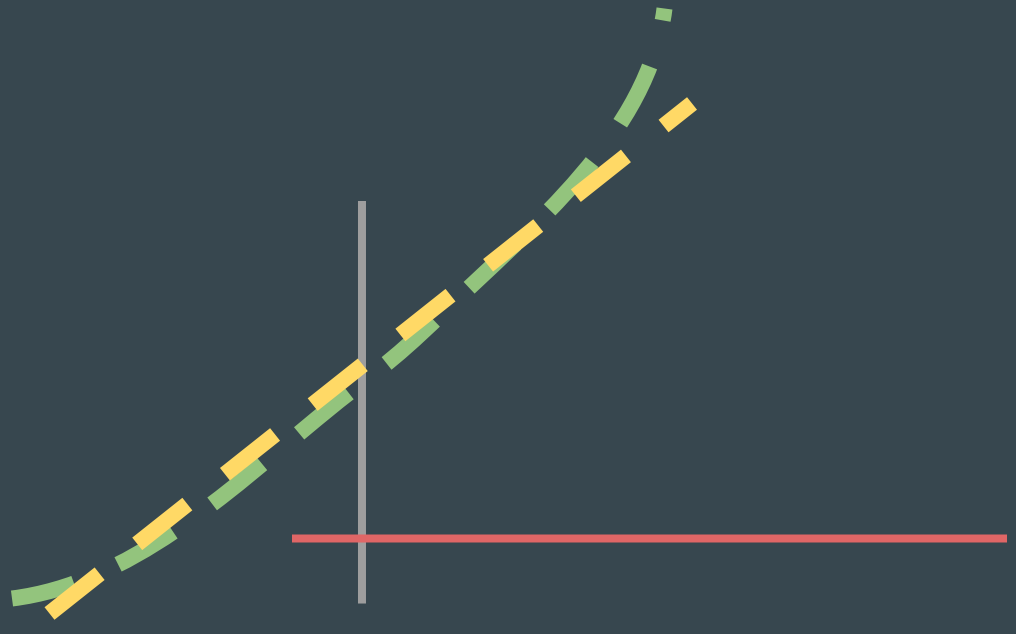
## Cons:

- Soooo slow.

# Frozen graph



$$x \cdot 0.95 + 1.05$$



$$x \cdot 0.95 + 1.05$$

Graph

Variables

Graph

Constants

# Frozen graph - Save

```
##### GRAPH #####
saver = tf.train.Saver()

##### SESSION #####
saver.save(...)
```

# Frozen graph - Save

```
##### GRAPH #####
saver = tf.train.Saver()

##### SESSION #####
saver.save(...)
```

.ckpt

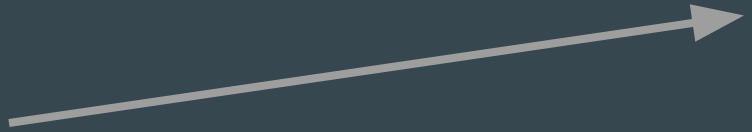
# Frozen graph - Save

```
##### GRAPH #####
saver = tf.train.Saver()

##### SESSION #####
saver.save(...)

tf.train.write_graph(...)
```

.ckpt

# Frozen graph - Save

```
##### GRAPH #####
saver = tf.train.Saver()

##### SESSION #####
saver.save(...)

tf.train.write_graph(...)
```



.ckpt

.pb

# Frozen graph - Freeze

graph.pb

.ckpt

# Frozen graph - Freeze

graph.pb

.ckpt

```
from tensorflow.python.tools import freeze_graph

freeze_graph.freeze_graph('graph.pb', ..., 'model.ckpt', ...)
```

# Frozen graph - Freeze

graph.pb

.ckpt

```
from tensorflow.python.tools import freeze_graph

freeze_graph.freeze_graph('graph.pb', ..., 'model.ckpt', ...)
```

frozen_graph.pb

# Frozen graph - Load

frozen_graph.pb

# Frozen graph - Load

frozen_graph.pb

```
tf.import_graph_def(graph_def, name='')
```

# Frozen graph - Load

## frozen_graph.pb

```
from tensorflow.core.framework import graph_pb2

graph_def = graph_pb2.GraphDef()



tf.import_graph_def(graph_def, name='')
```

# Frozen graph - Load

frozen_graph.pb

```
from tensorflow.core.framework import graph_pb2

graph_def = graph_pb2.GraphDef()

with open(‘frozen_graph.pb’, ‘rb’) as f:

    graph_def.ParseFromString(f.read())

tf.import_graph_def(graph_def, name='')
```

running is the same

# Frozen Graph - Summary

<table>
<tr><th>Save</th><th>Files</th><th>Freeze</th><th>File</th><th>Load</th></tr>
<tr><td>tf.train.write_graph()</td><td>.pb</td><td rowspan="2">freeze_<br>graph()</td><td rowspan="2">.pb</td><td rowspan="2">graph_def.ParseFromString()<br><br>tf.import_graph_def(graph_def)</td></tr>
<tr><td>tf.train.Saver -><br>.save()</td><td>.ckpt</td></tr>
</table>

# Compiled

# Compiled

## Pros:

- Very fast.
- Just 60 Mb (executable) + 17 Mb (frozen model) with no dependencies.

## Cons:

- Need to be built in C++ with Bazel...

demo

# Retraining using checkpoints

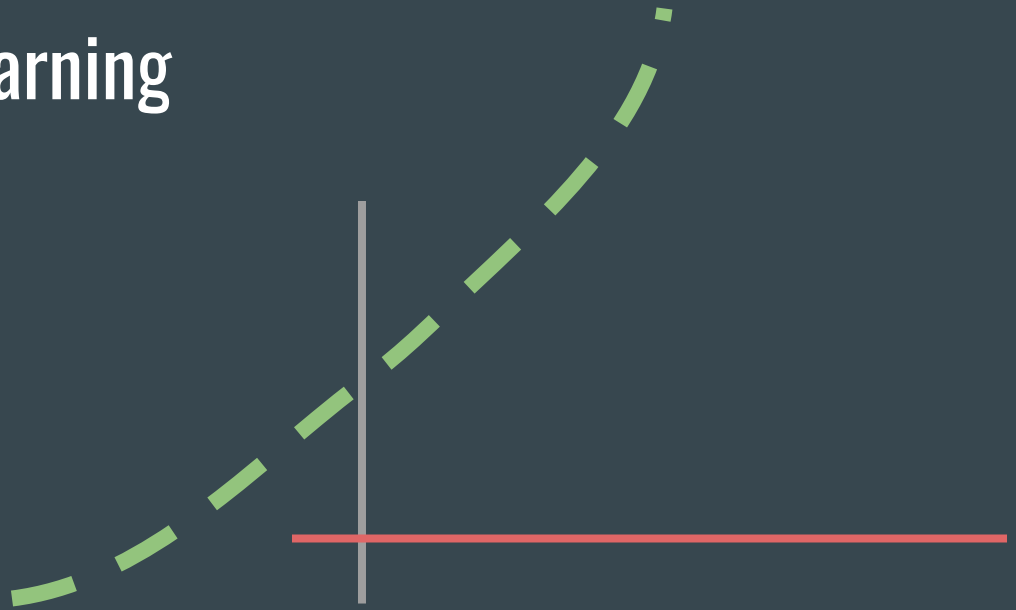
# Fine-tuning

```
targets = tf.placeholder(tf.float32, shape=[None, n_classes])

# define a cost function and optimizer for these targets


with tf.Session() as sess:
    saver.restore(sess, 'model.ckpt')
    sess.run(optimizer, feed_dict={
        inputs: inputs,
        targets: targets
        })
```

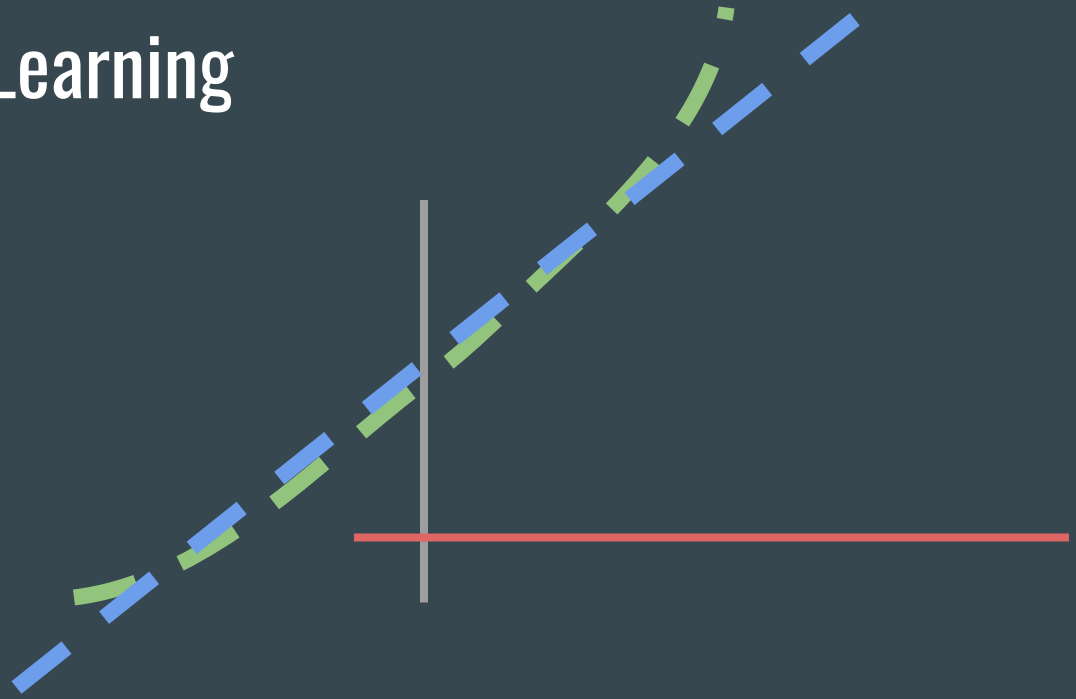
# Continue training

```
optimizer =
tf.get_default_graph().get_tensor_by_name('name:0')

# same for inputs and targets


with tf.Session() as sess:
    saver.restore(sess, ‘model.ckpt’)
    sess.run(optimizer, feed_dict={
        inputs: inputs,
        targets: targets
        })
```

# Transfer Learning

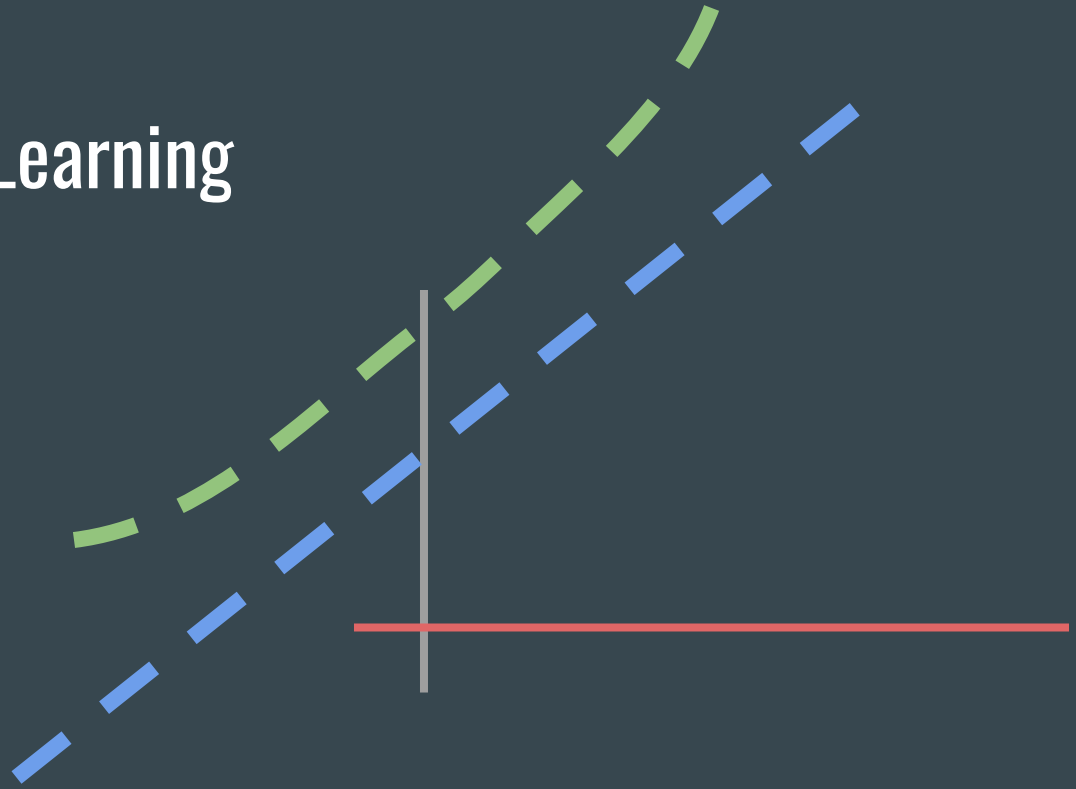


# Transfer Learning



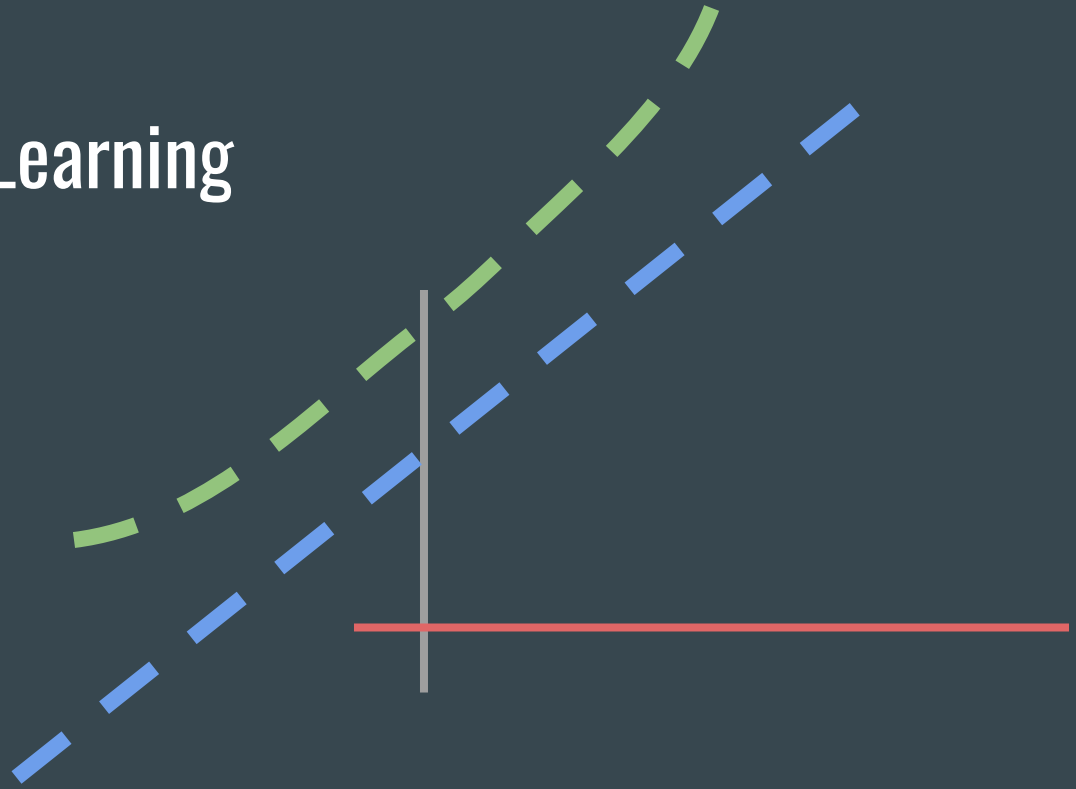
$$x \cdot 0.95 + 1.05$$

# Transfer Learning



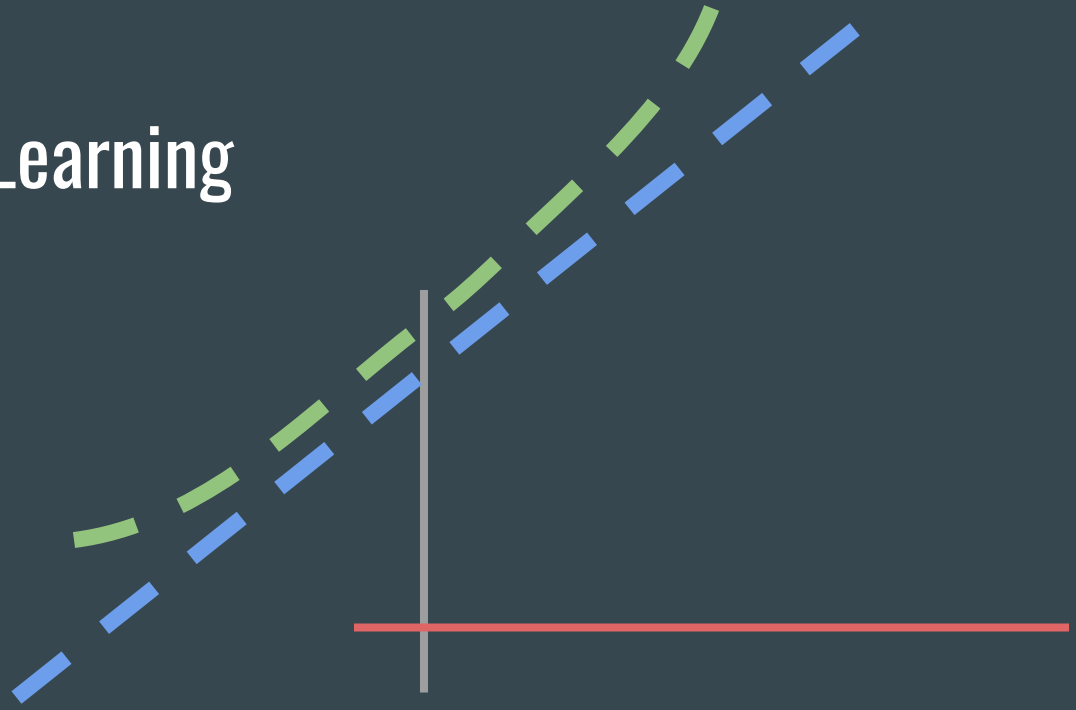
$$x \cdot 0.95 + 1.05$$

# Transfer Learning



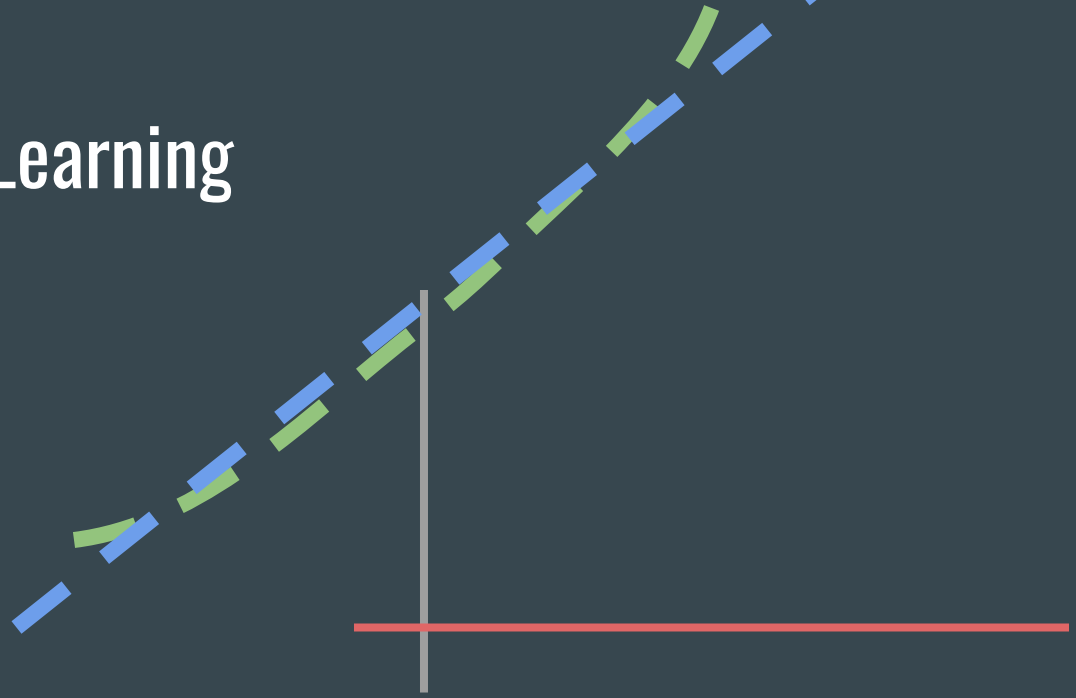
$$x \cdot 0.95 + 1.05$$

# Transfer Learning



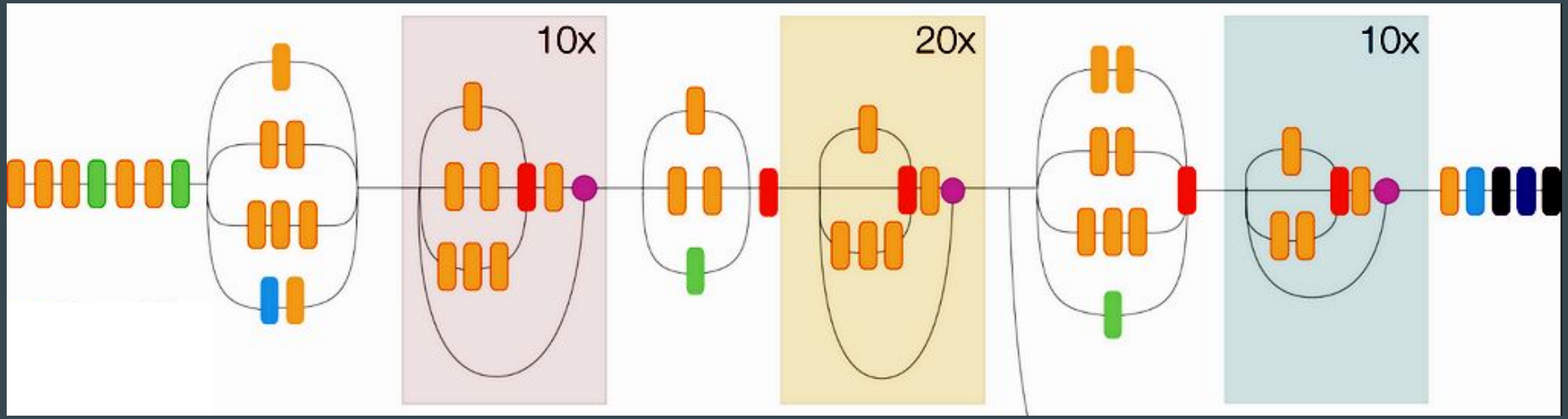
$$x \cdot 0.95 + 1.20$$

# Transfer Learning

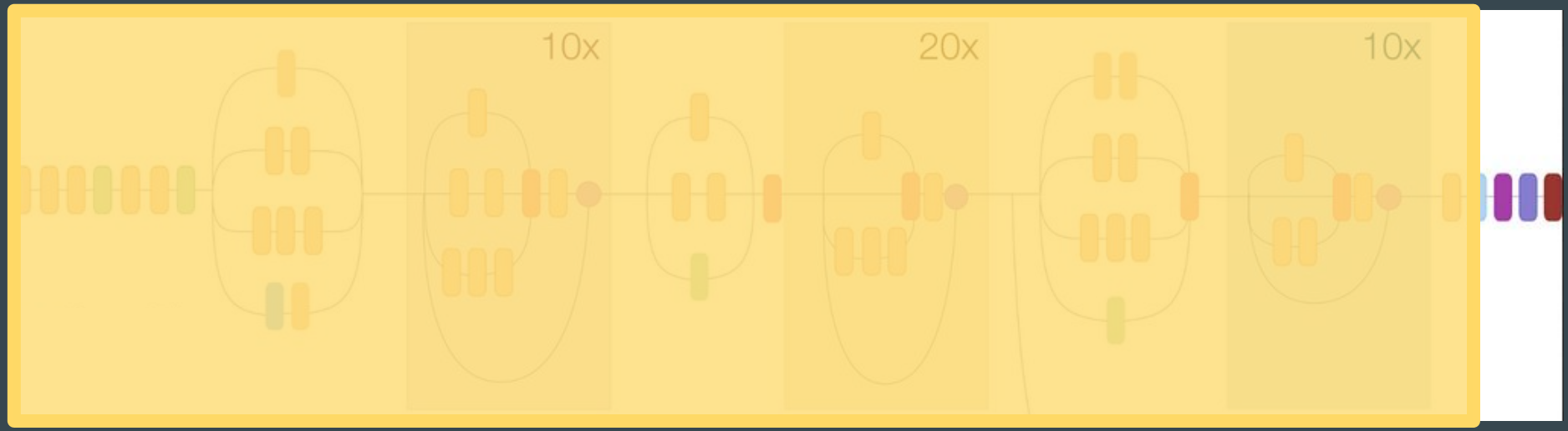


$$x \cdot 0.95 + 1.32$$

# Transfer Learning

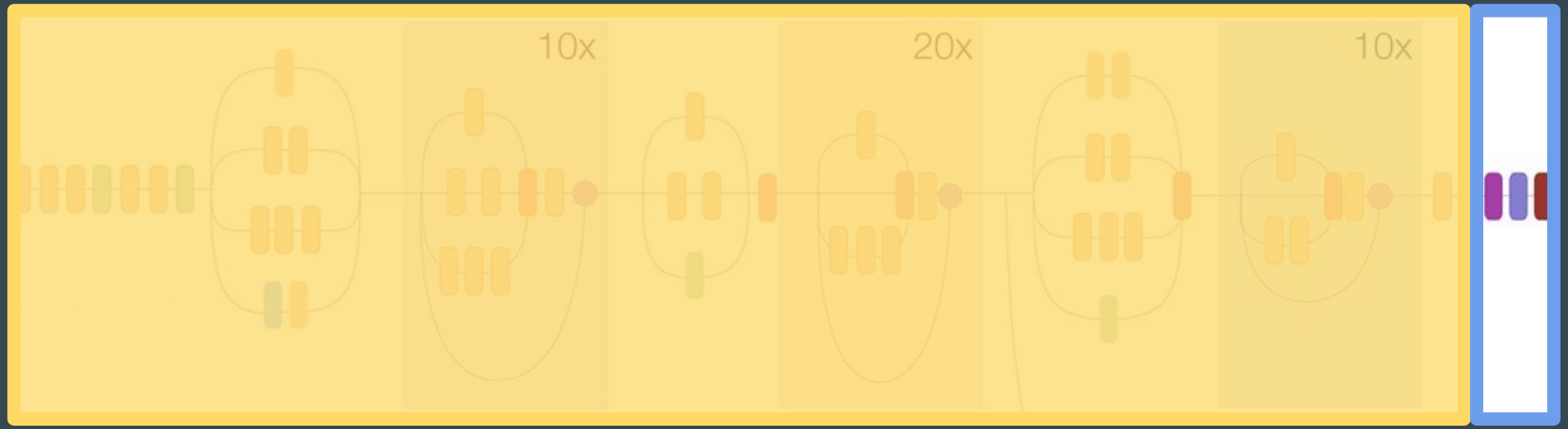


# Transfer Learning




static

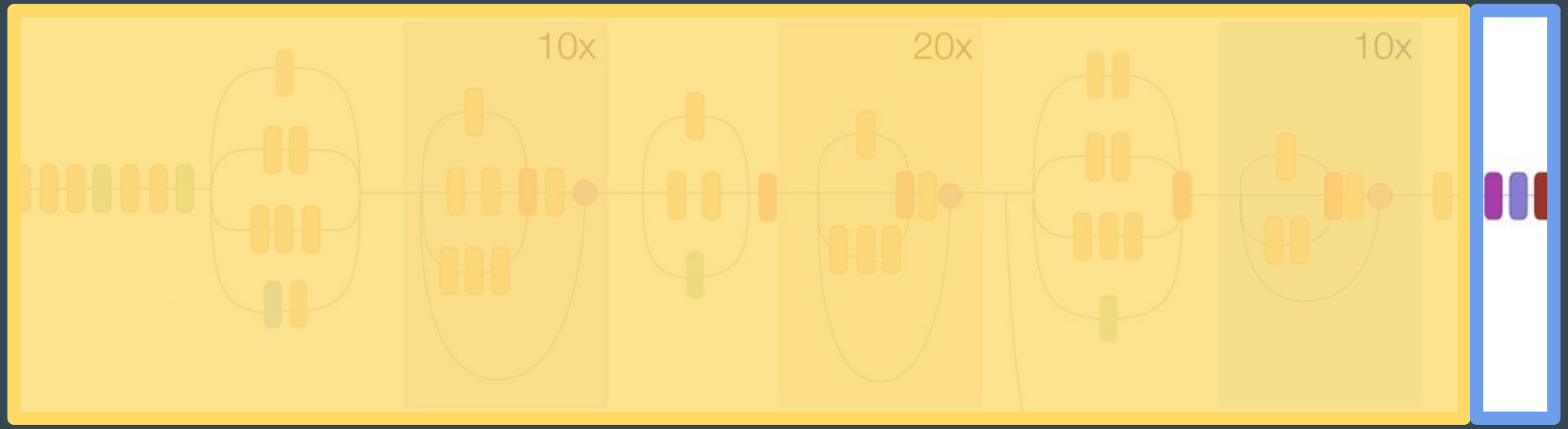
# Transfer Learning




static

dynamic

# Transfer Learning

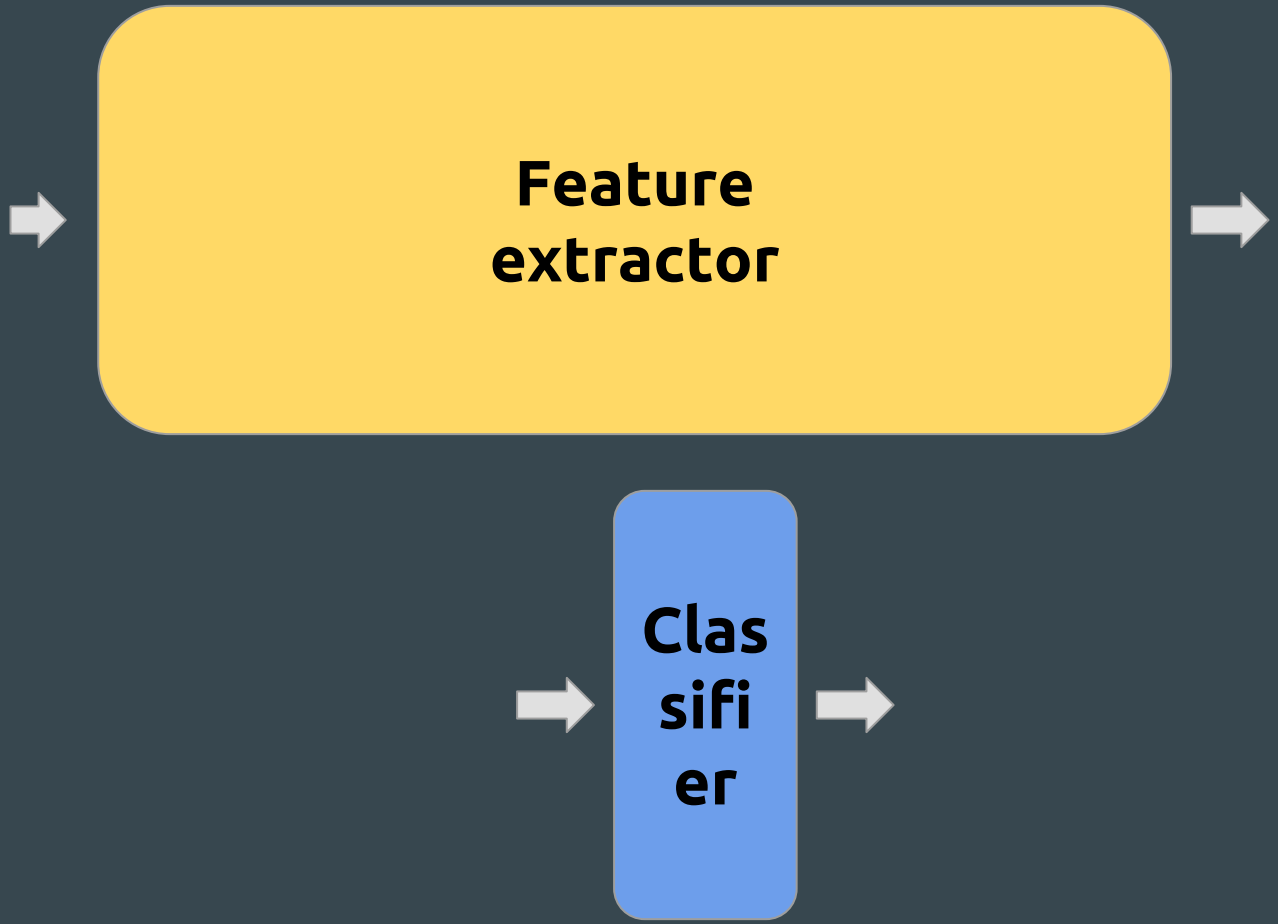


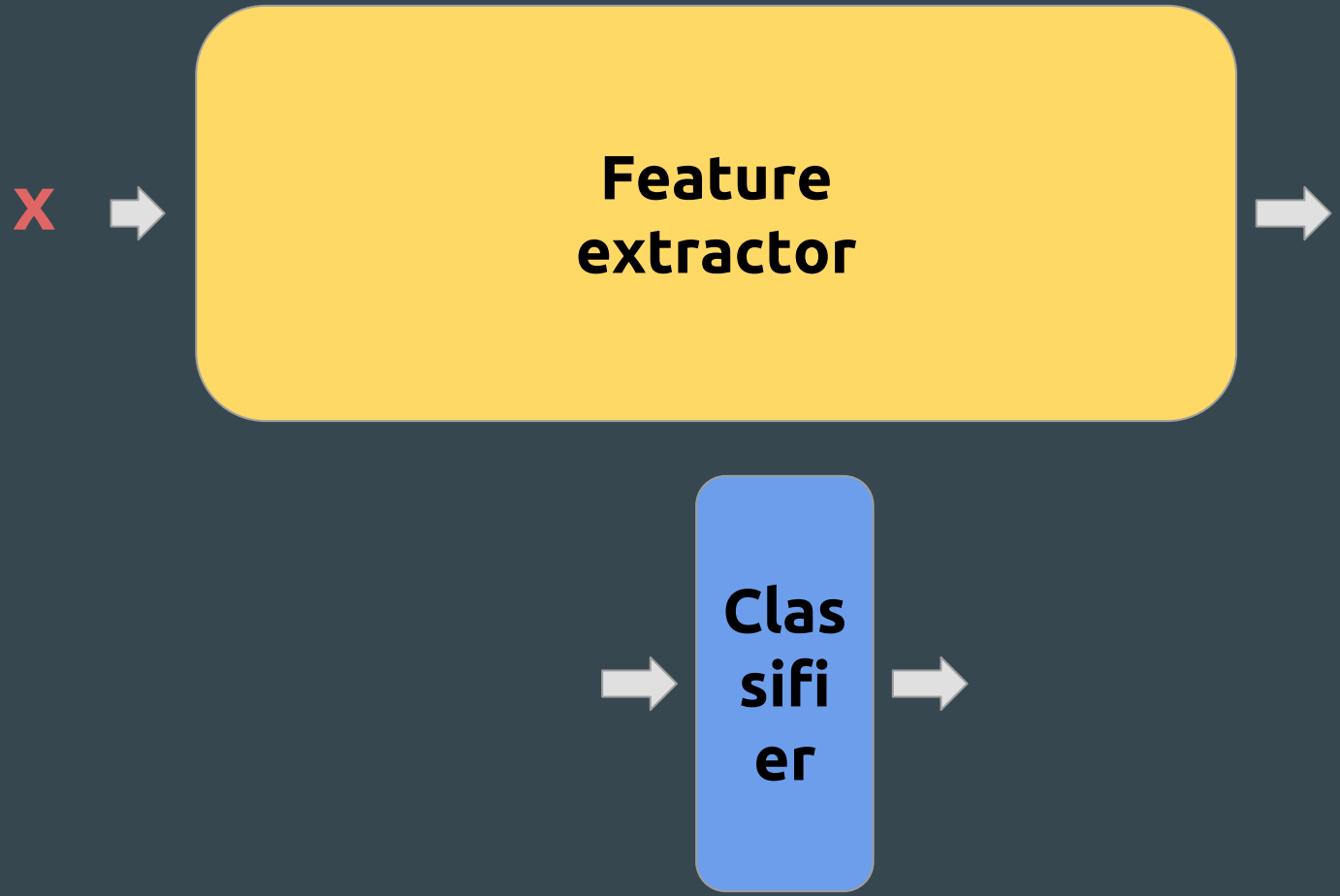

bottleneck

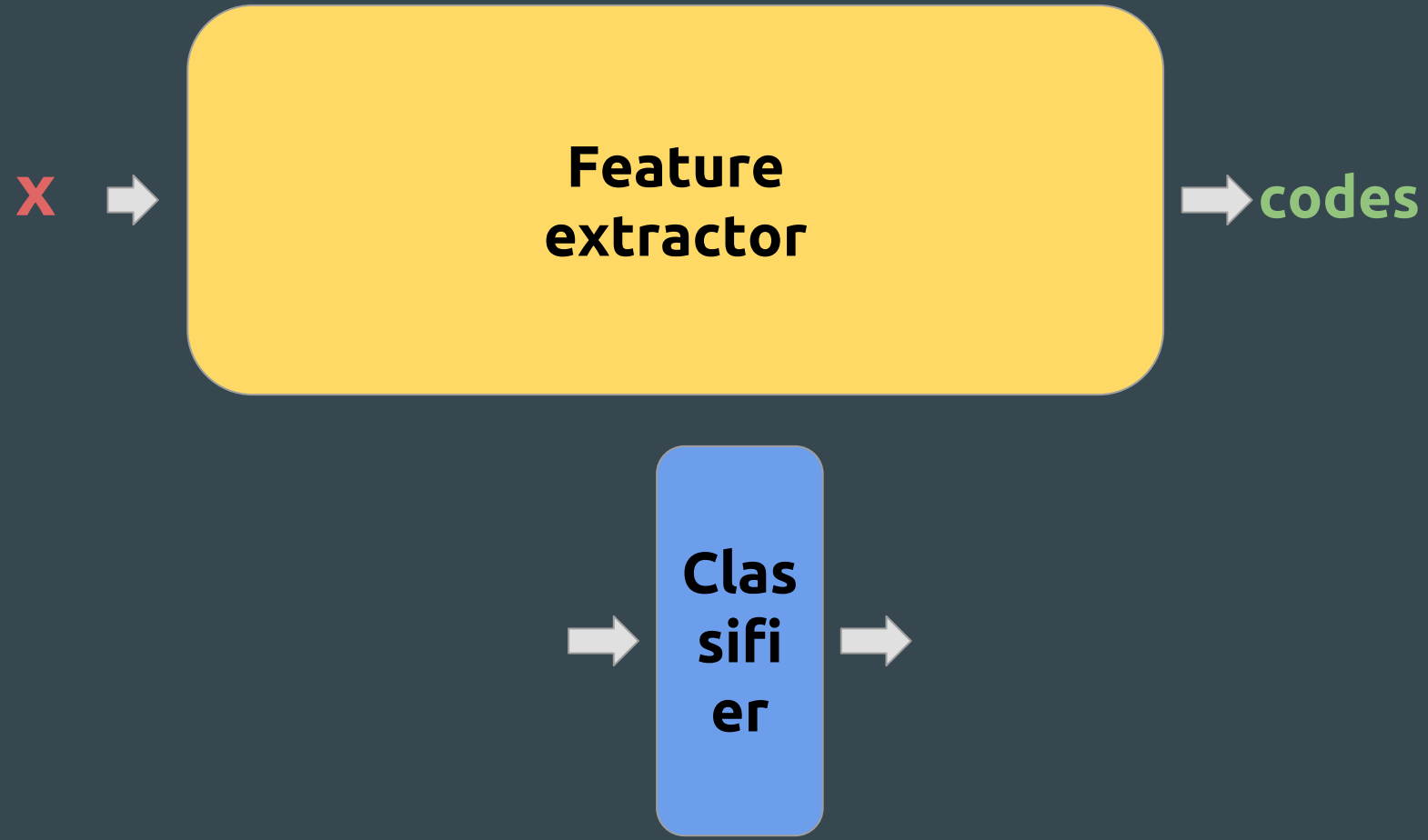
static

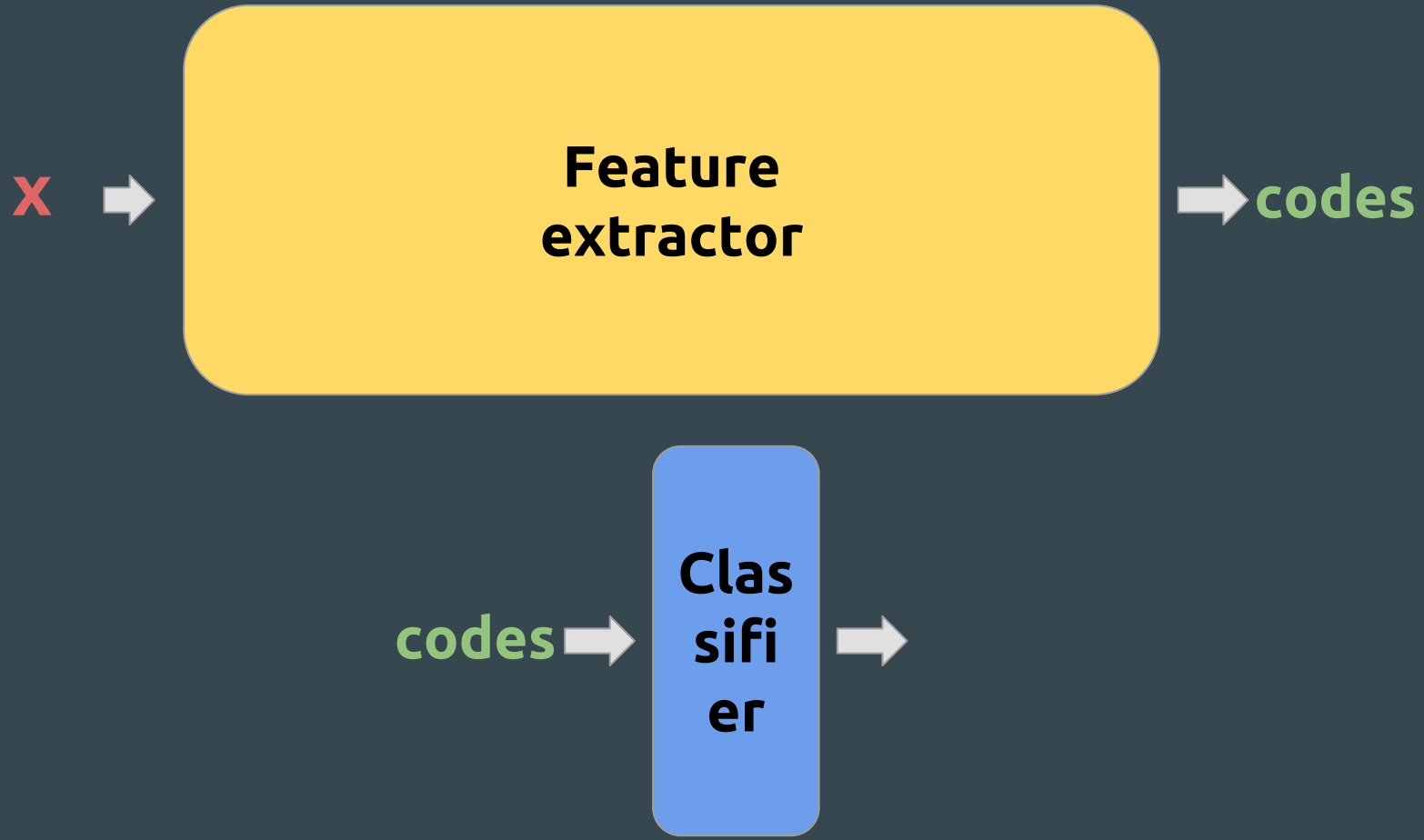
dynamic

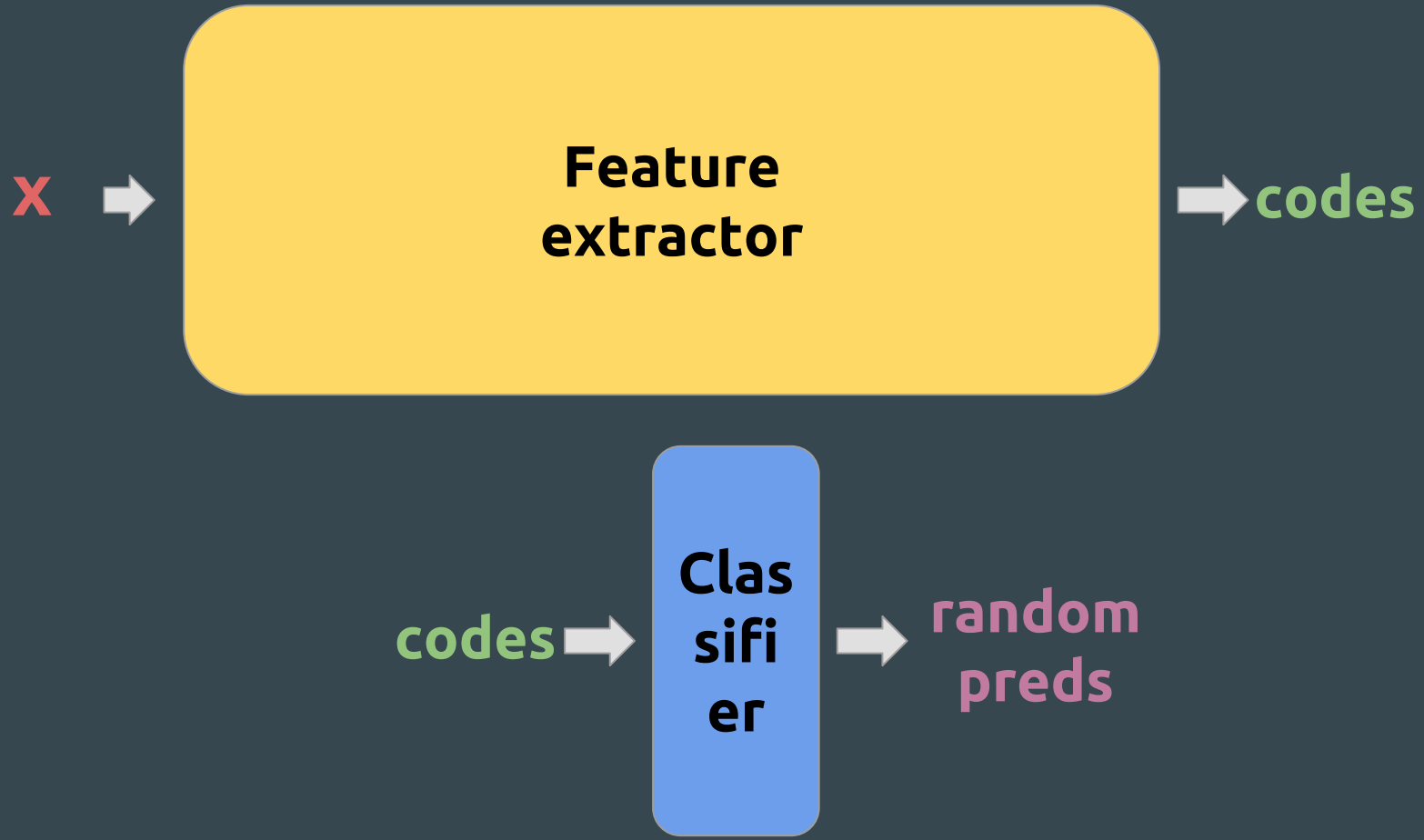
# Transfer Learning

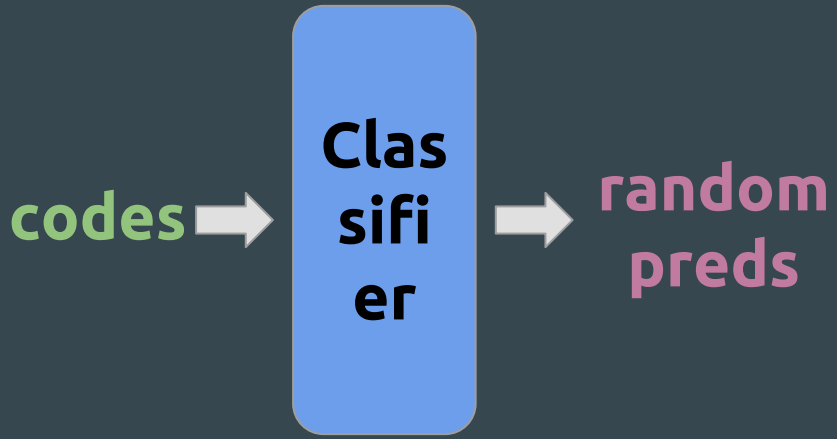

Feature
extractor
Clas
sifi
er

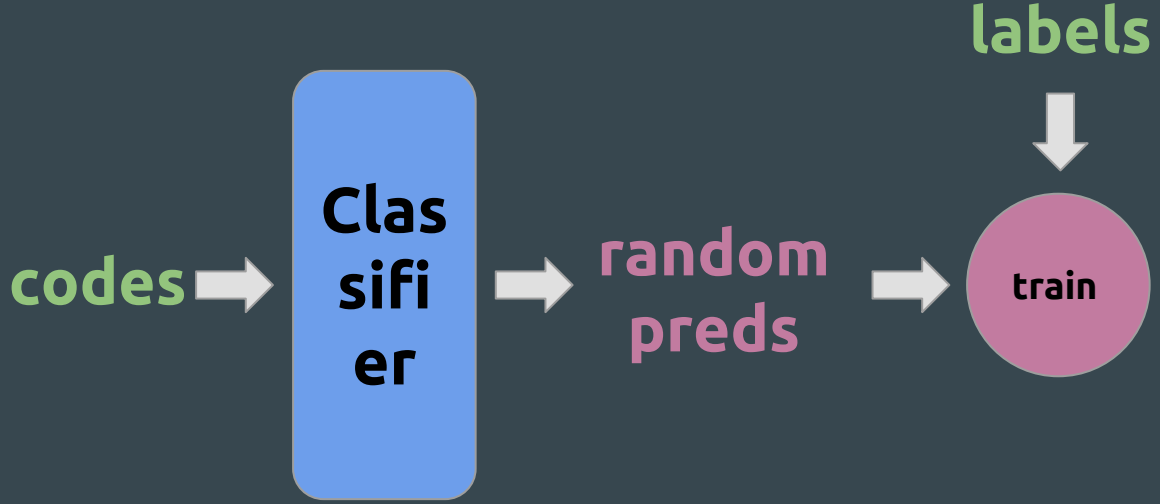

x
Feature extractor
Clas sifi er

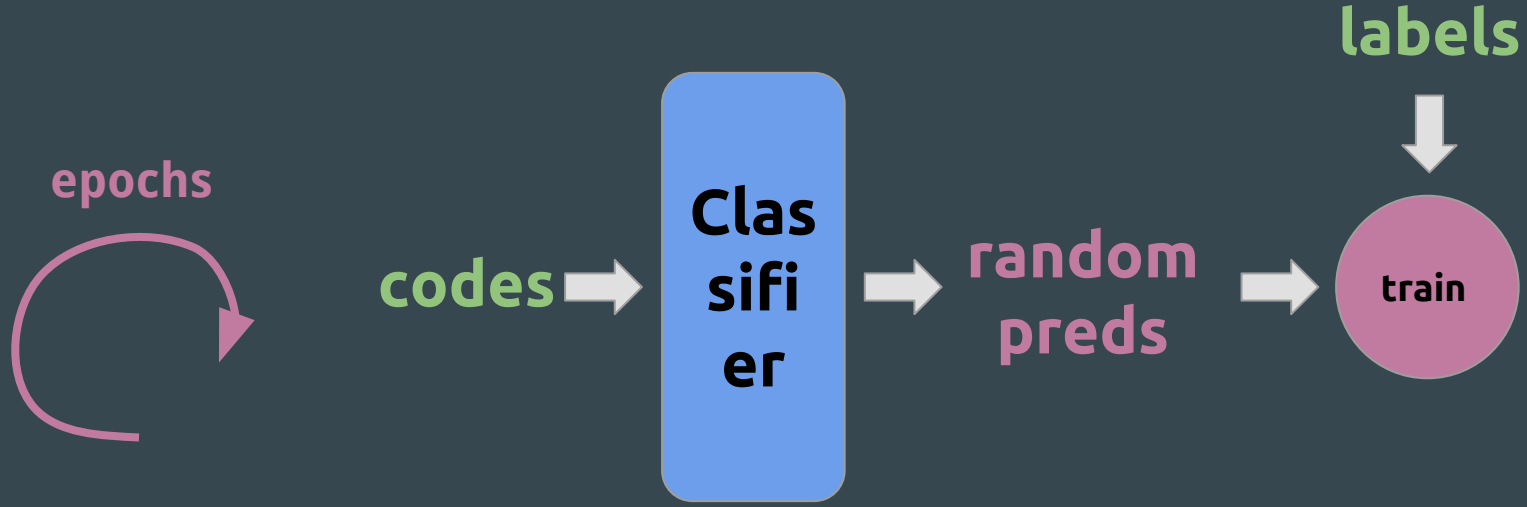

x
Feature extractor
codes
Clas sifi er


x
Feature extractor
codes
codes
Clas sifi er


x
Feature extractor
codes
codes
Clas sifi er
random preds


codes
Clas
sifi
er
random
preds


labels
codes
Clas
sifi
er
random
preds
train


epochs
codes
Clas
sifi
er
random
preds
labels
train

# Extract features

```
with tf.Session() as sess:
    code = sess.run(bottleneck, feed_dict={
        input_placeholder: image
        })
```

# Extract features

```
bottleneck =
tf.get_default_graph().get_tensor_by_name('name:0')


with tf.Session() as sess:
    code = sess.run(bottleneck, feed_dict={
        input_placeholder: image
        })
```

# Extract features

```
bottleneck =
tf.get_default_graph().get_tensor_by_name('name:0')


with tf.Session() as sess:
    code = sess.run(bottleneck, feed_dict={
        input_placeholder: image
        })
```

# Retrain classifier

```
with tf.Session() as sess:
    for epoch in range(len(epochs)):
        sess.run(optimizer, feed_dict={
            inputs_: codes
            lables_: labels
        })
```

hopefully, demo


I run on
COFFEE &
KNIGHTSCOUT
Jacob
Arnould

alesolano/imagenet_models_flask

alesolano/transfer_learning_webapp

alesolano/imagenet_models_flask

alesolano/transfer_learning_webapp

Thank you!