BC data Workshop BCSA

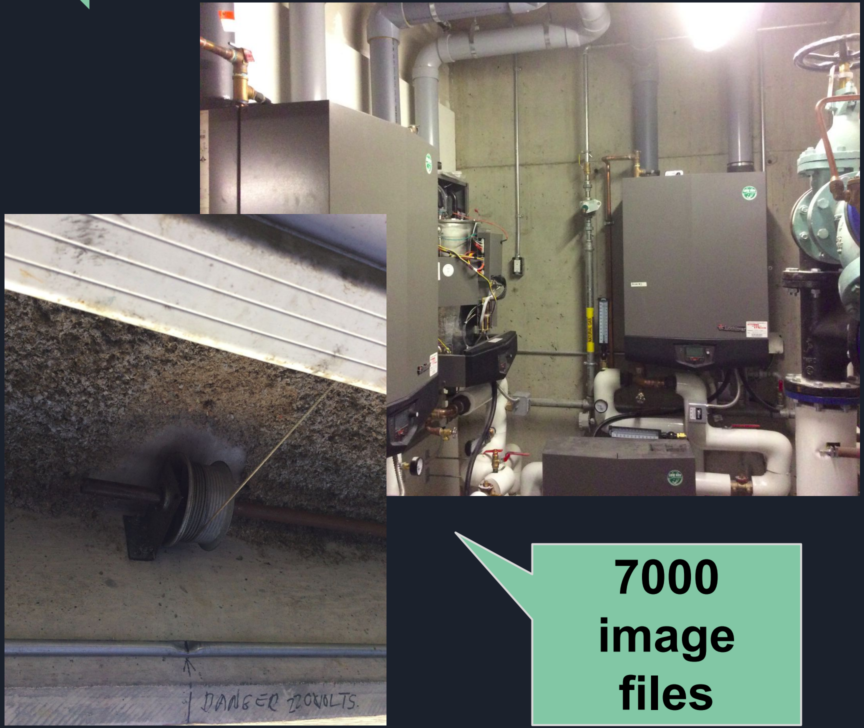
Group members:
Javier Hernandez, Alastair Jamieson-Lane, Jie Jian, Hans Oeri, Yi Sui, Shanzhao Wang

The Data

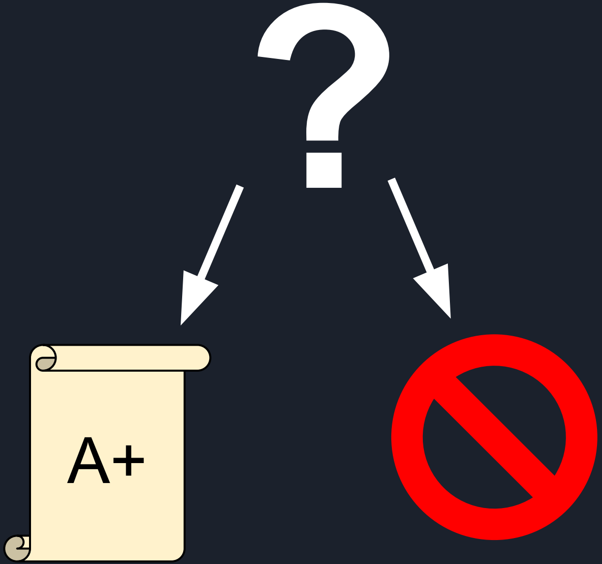
The Data

7000 image files

The Data



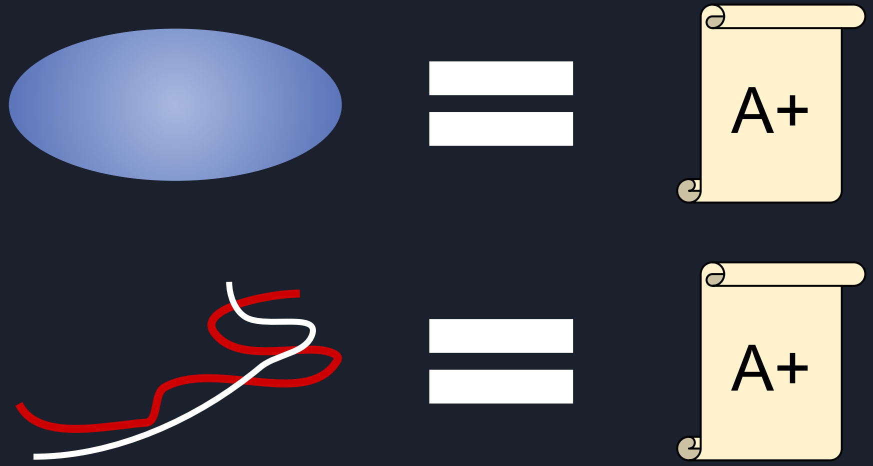
7000 image files



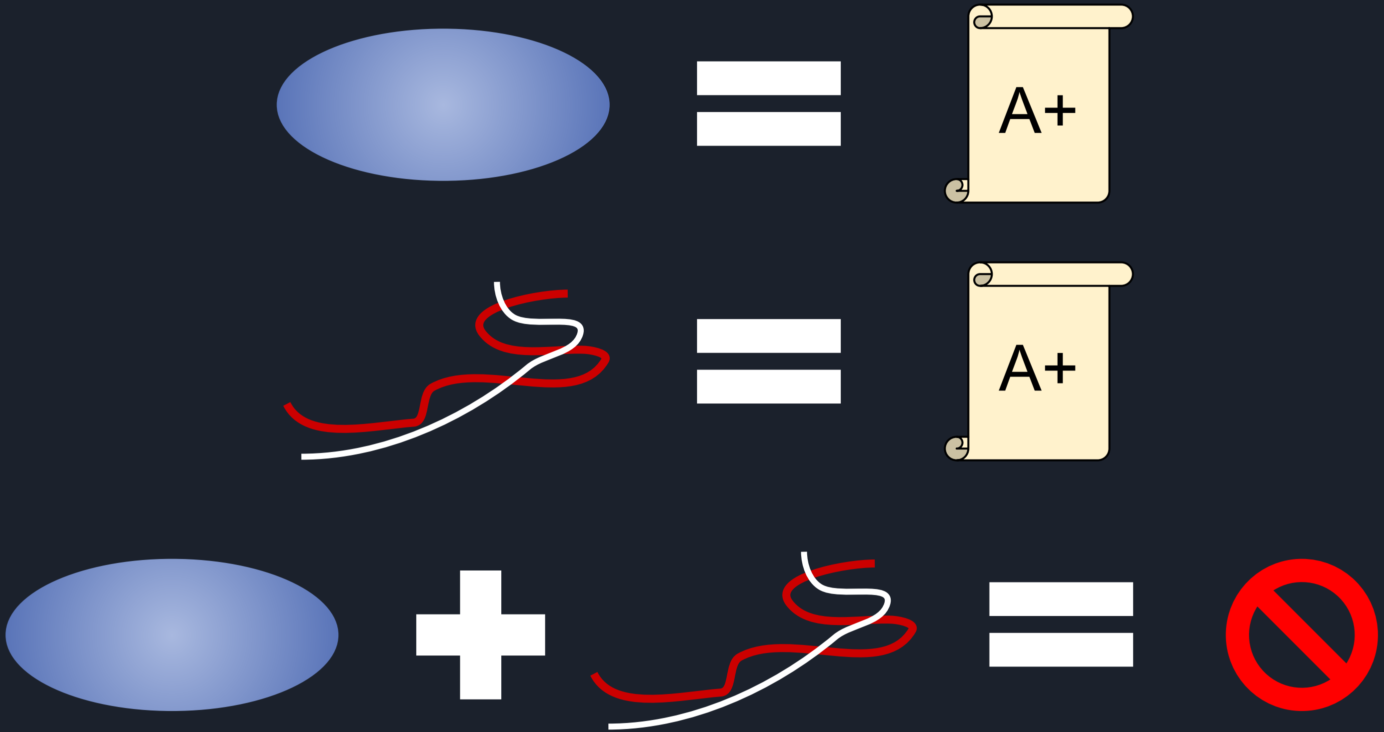
Challenges: Context

Challenges: Context

Challenges: Context



Challenges: Context

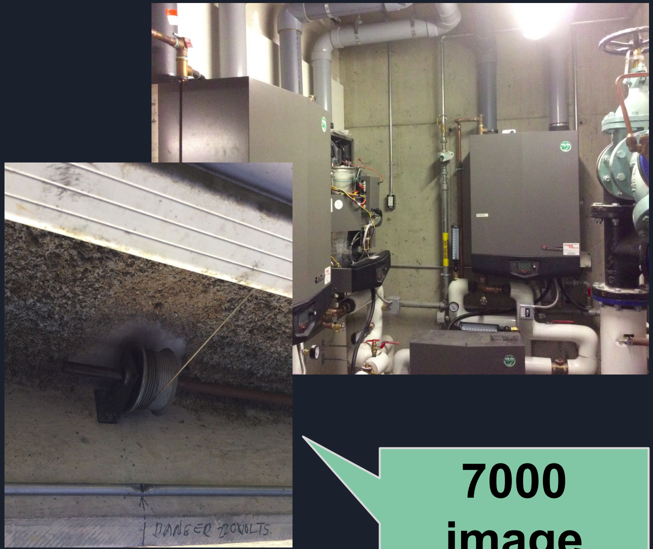


Challenges: Not many images

Challenges: Not many images

7000 image files

Challenges: Not many images



7000 image files

14,197,122 image files

Challenges: Not many images

VGG16

InceptionV3

ResNet

~100 million parameters

Transfer Learning

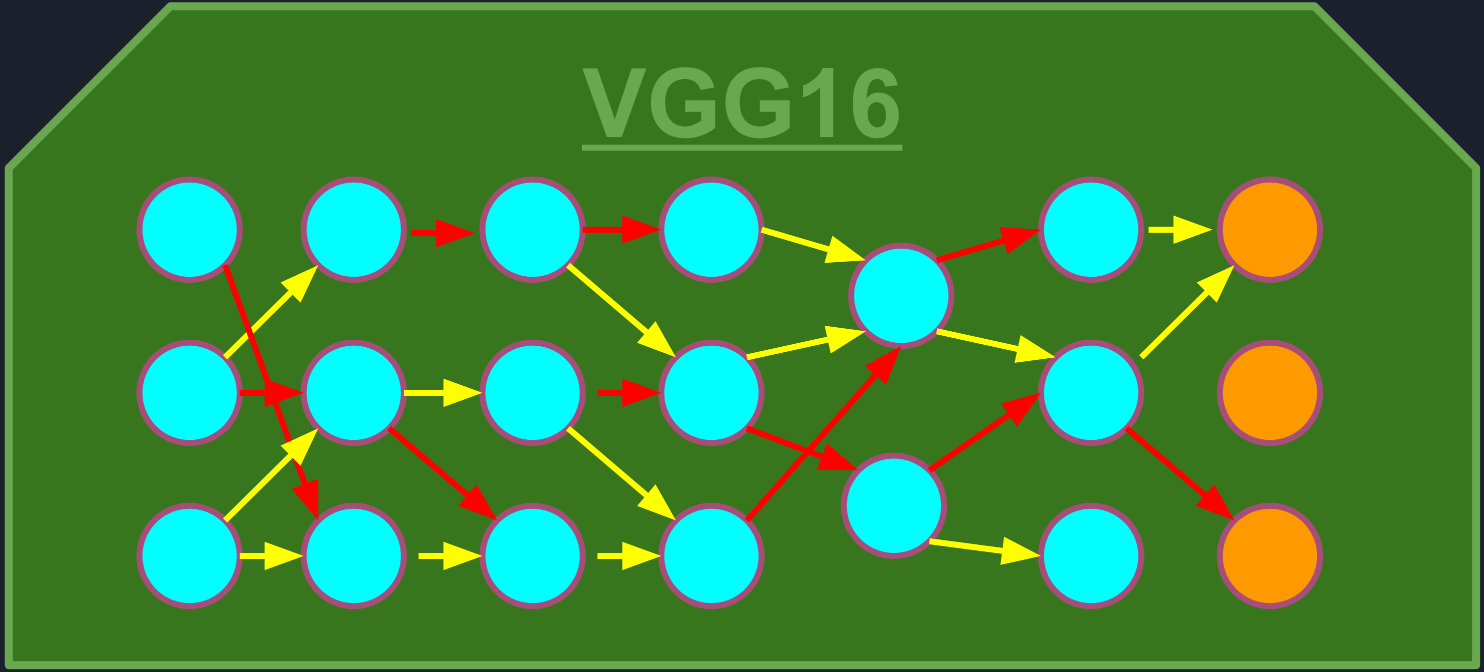
Transfer Learning

VGG16

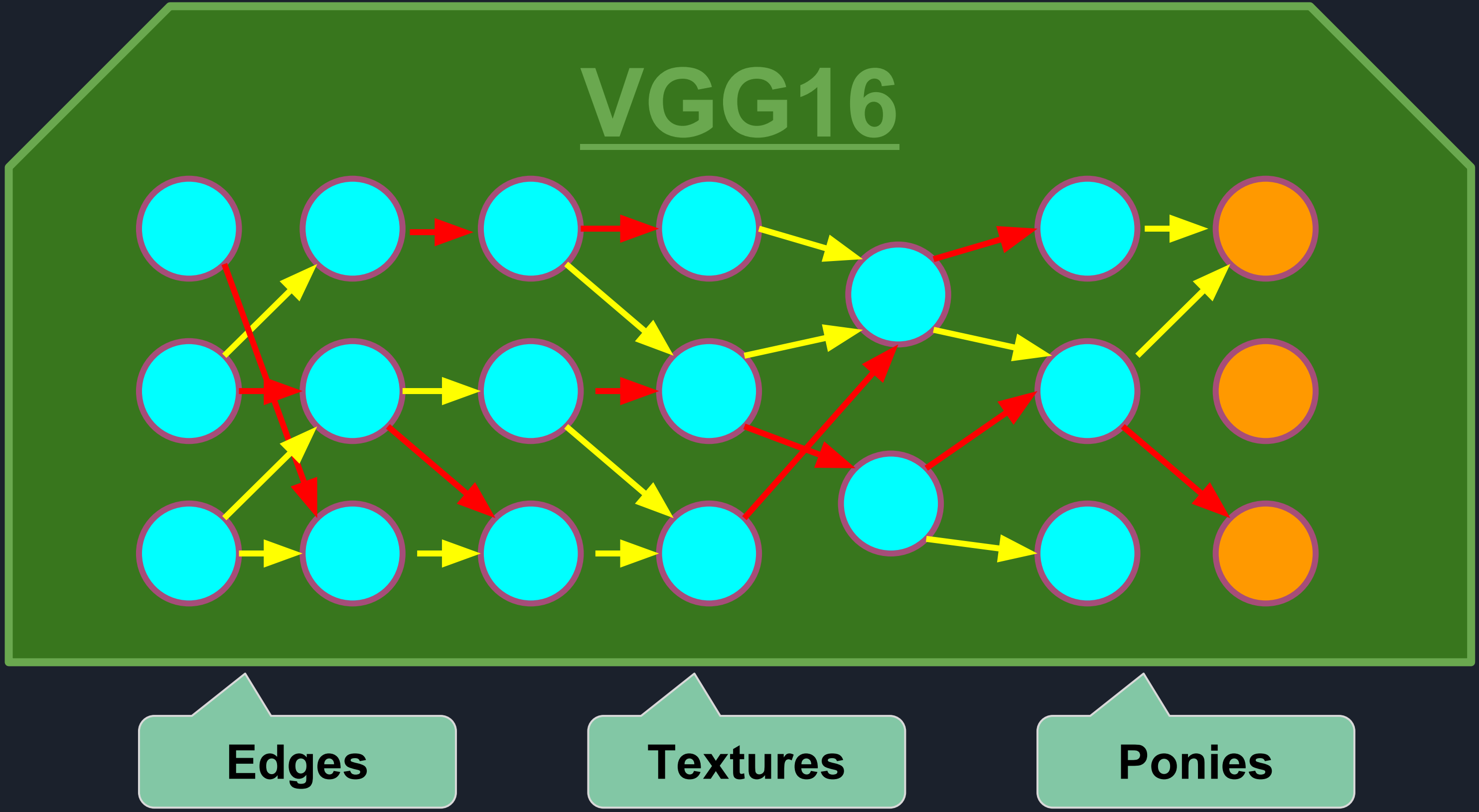
Transfer Learning

VGG16

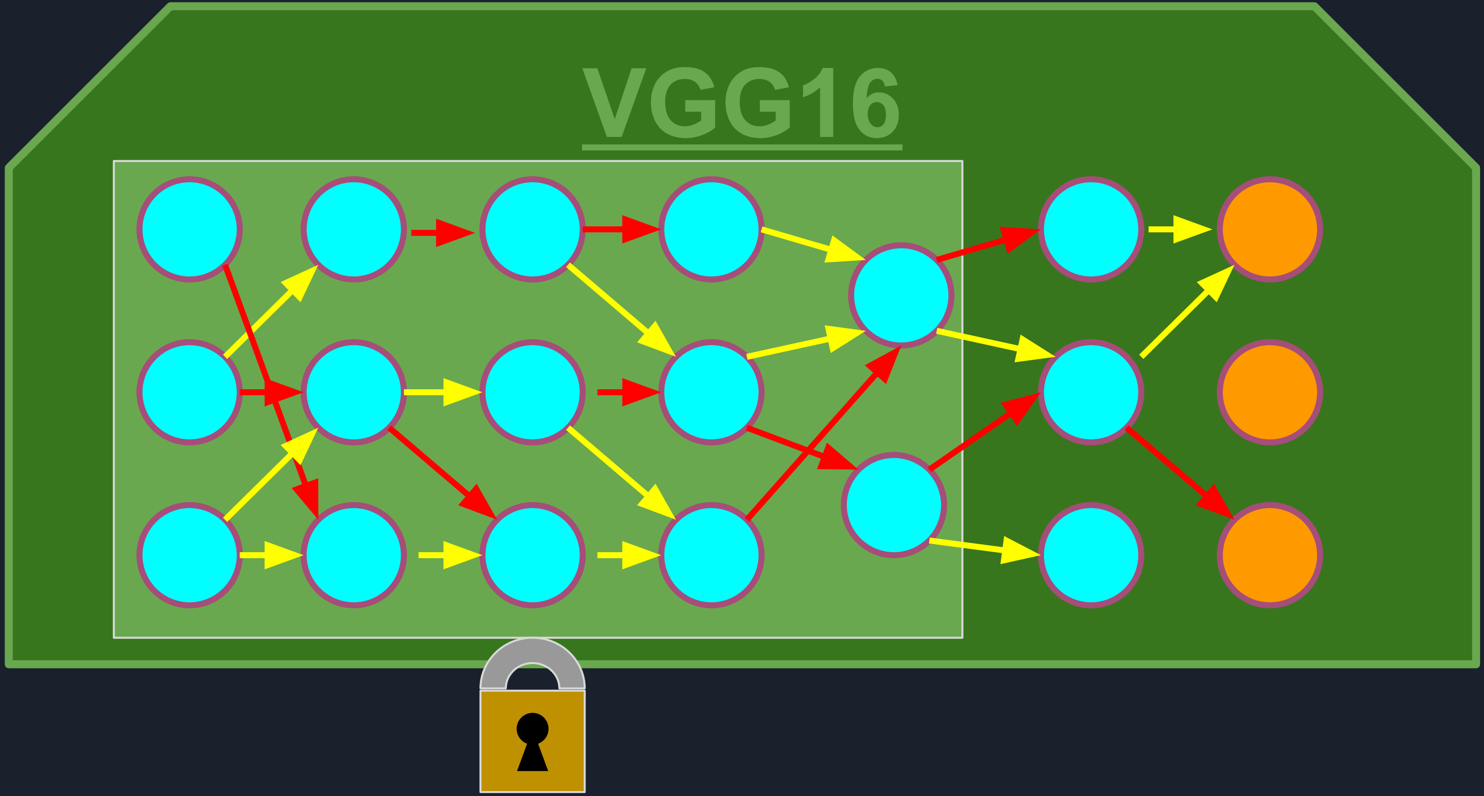
Transfer Learning



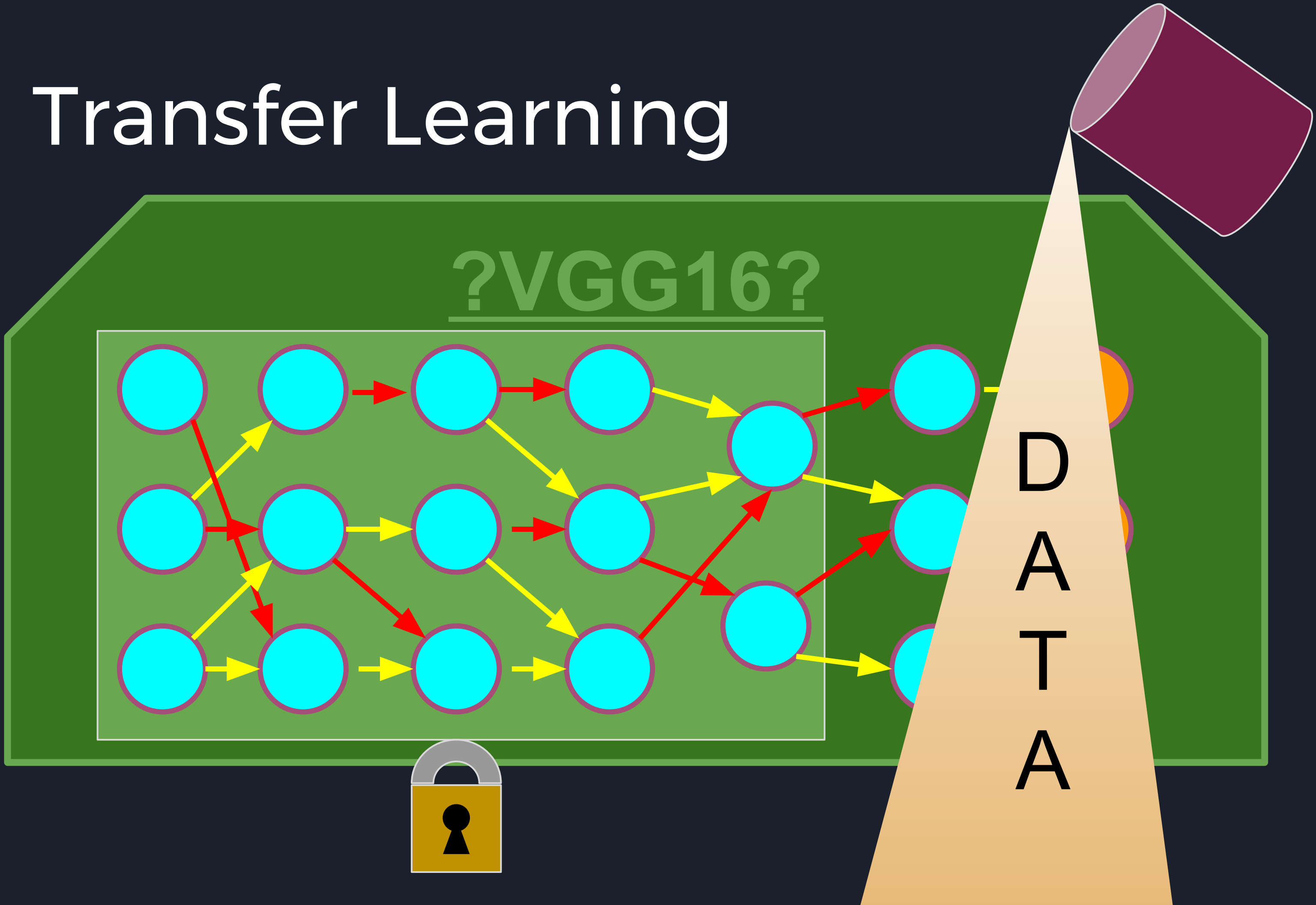
Transfer Learning



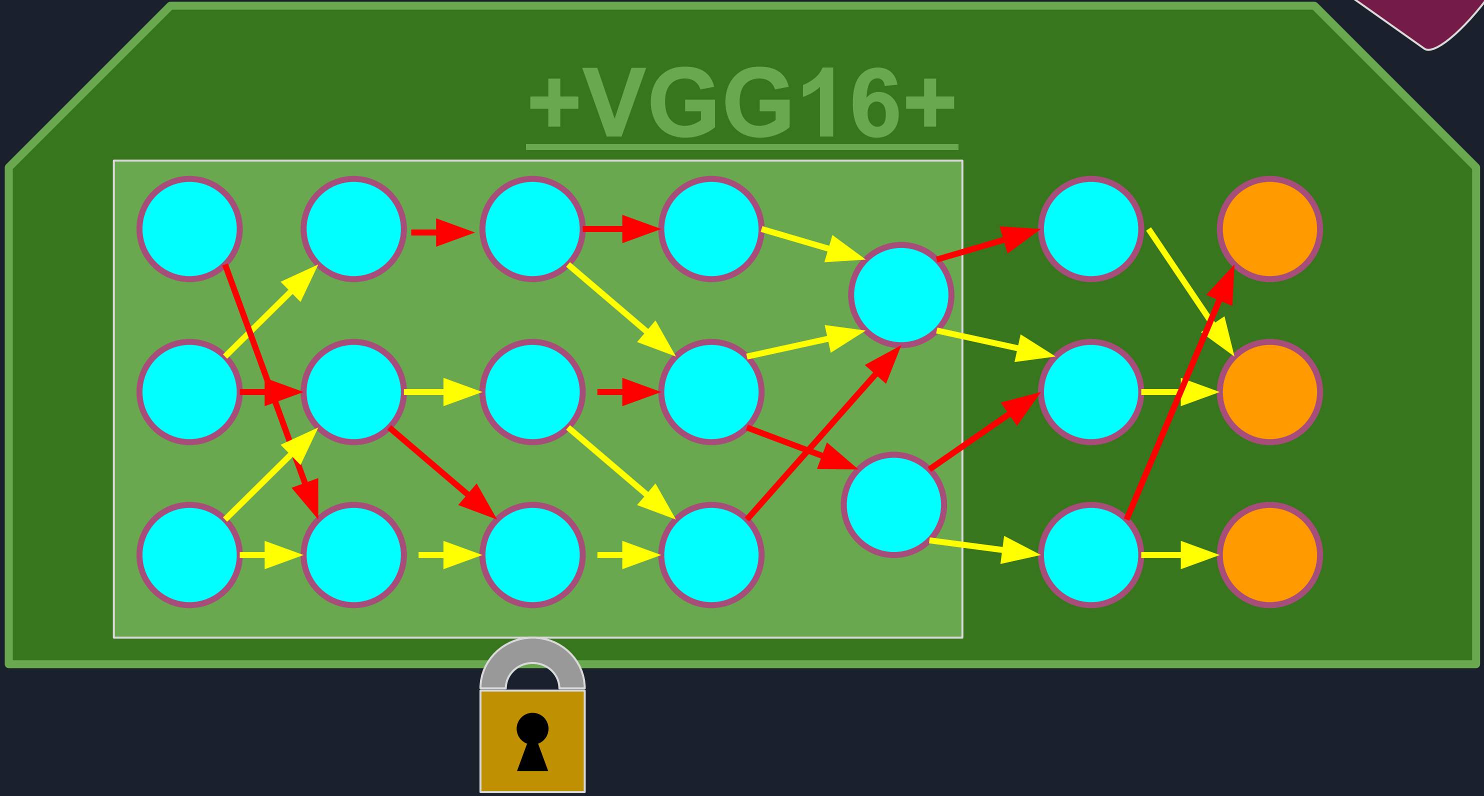
Transfer Learning



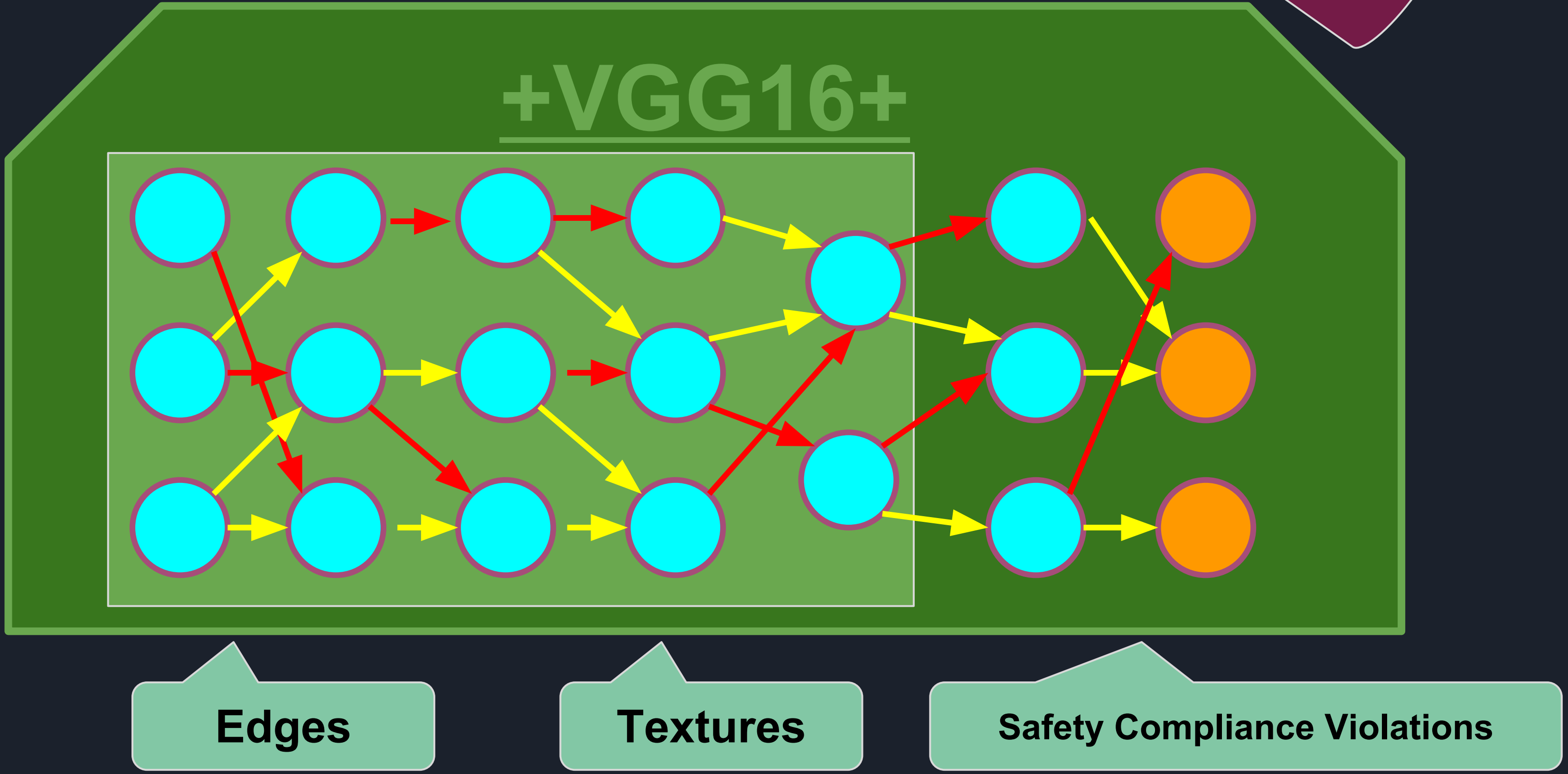
Transfer Learning



Transfer Learning



Transfer Learning



Challenges

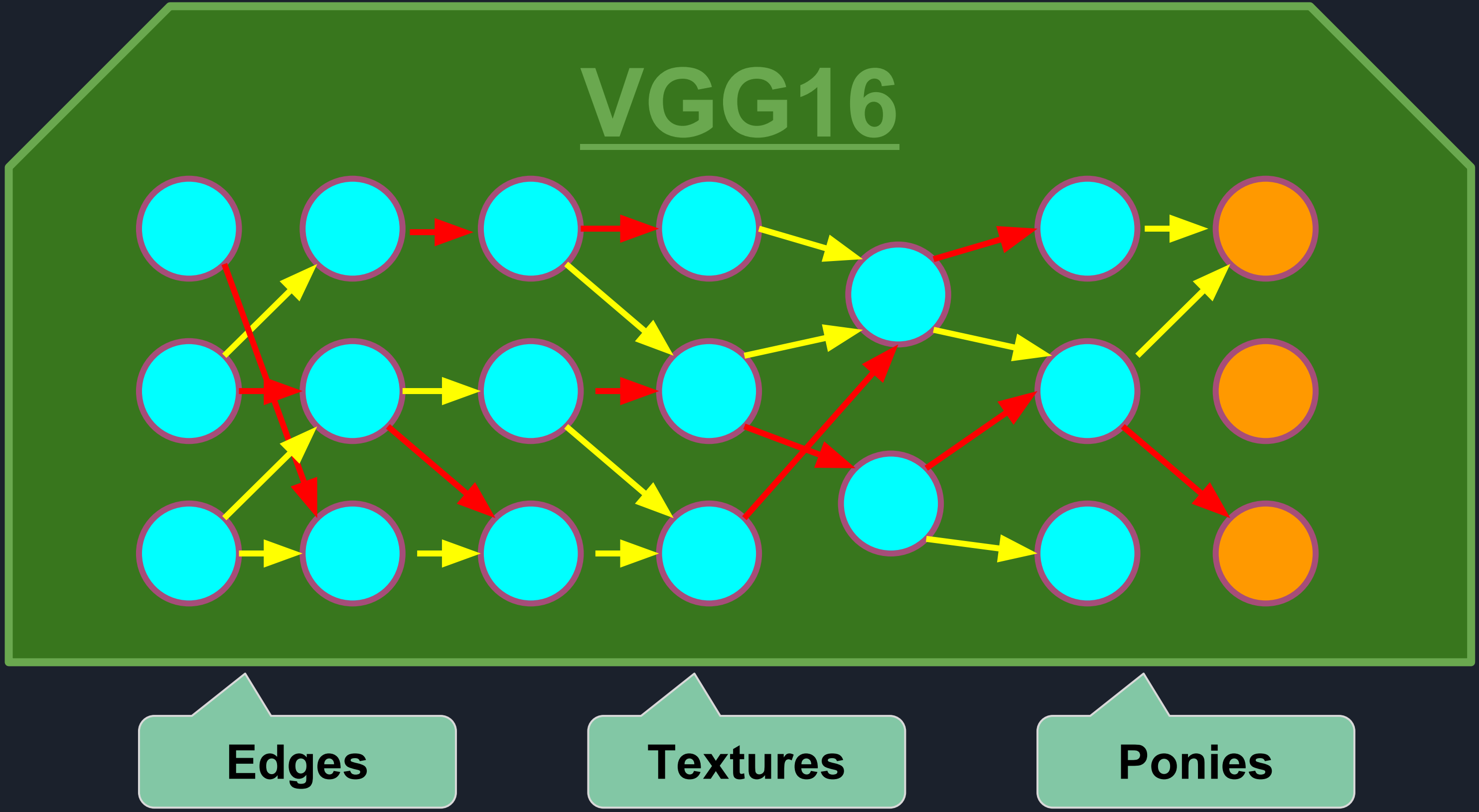
The data set we have to train on is relatively small for the degree of abstraction we have.

Despite small data sample, the storage needed is still very large and difficult to handle.

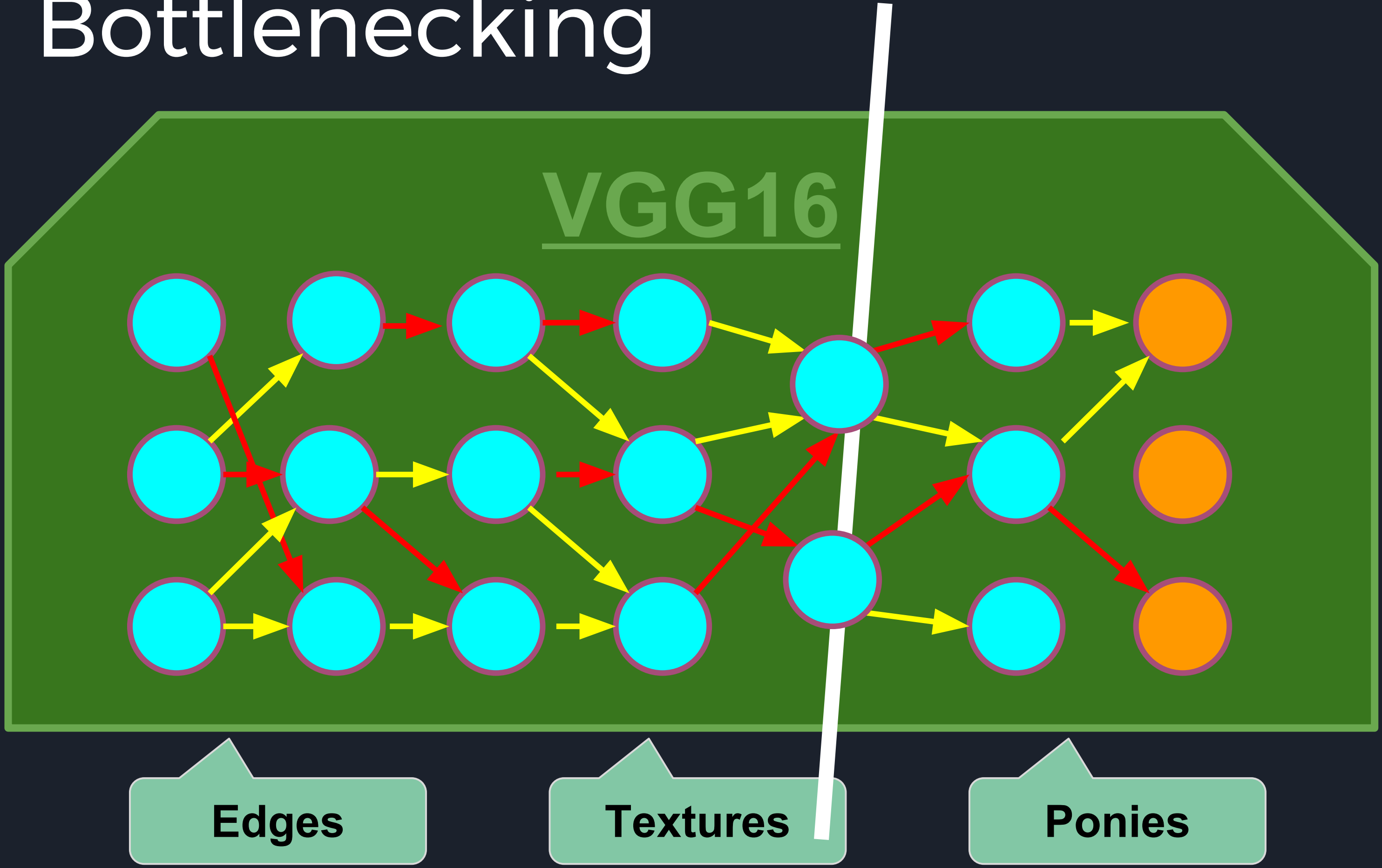
Training networks with many layers is time consuming.

Bottlenecking

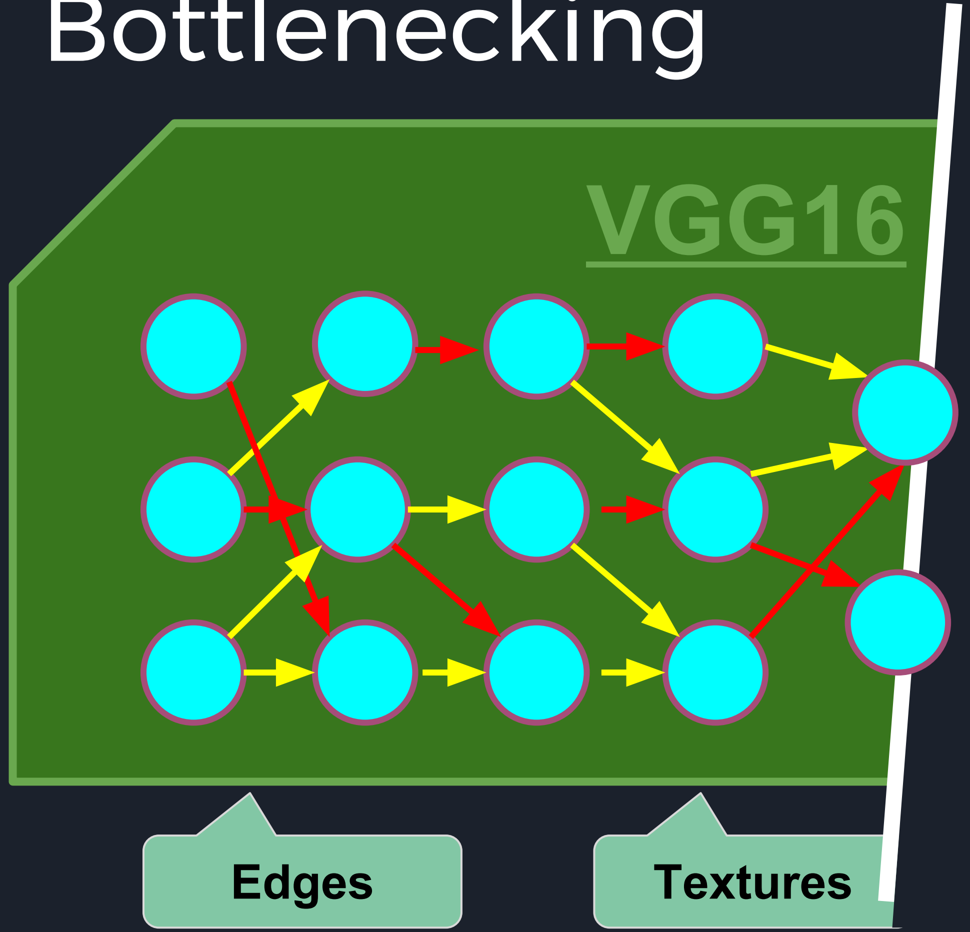
Bottlenecking



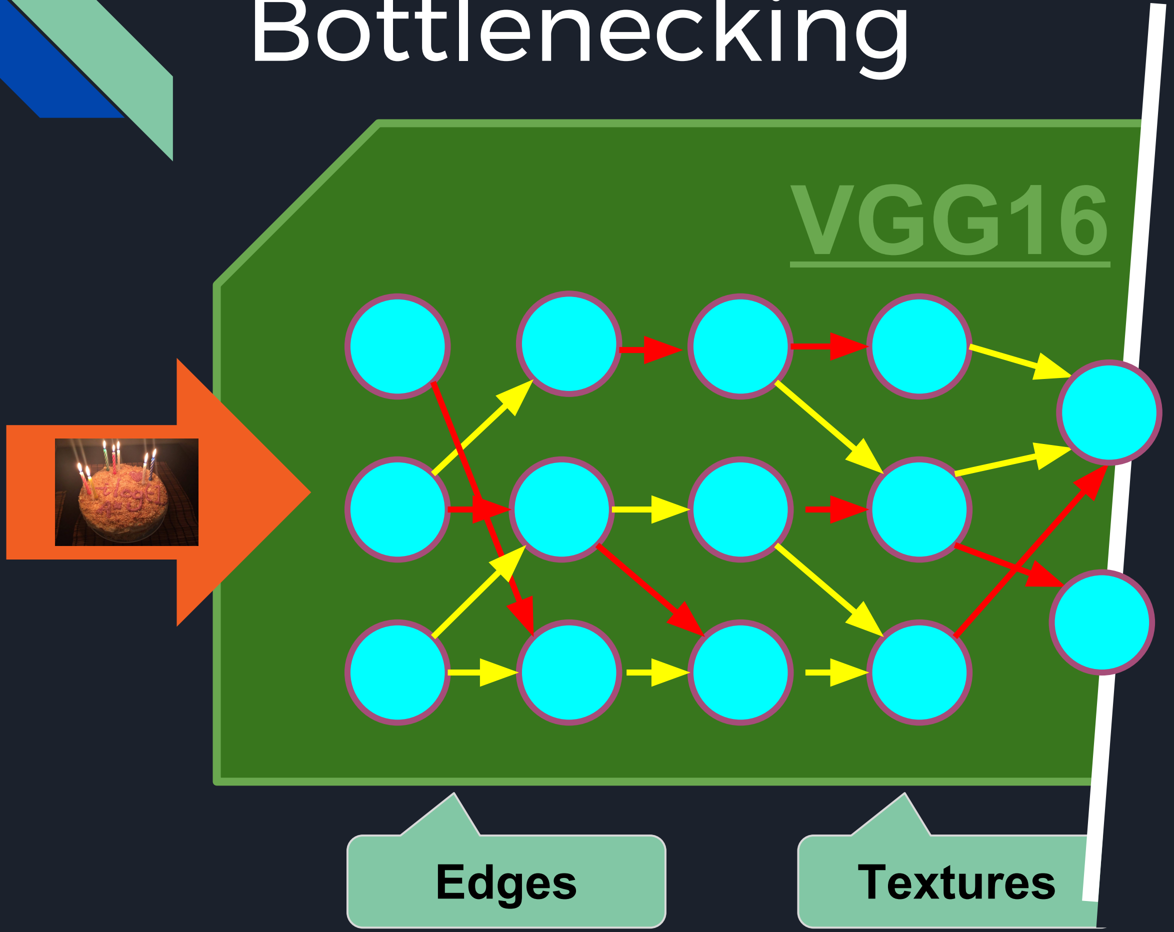
Bottlenecking



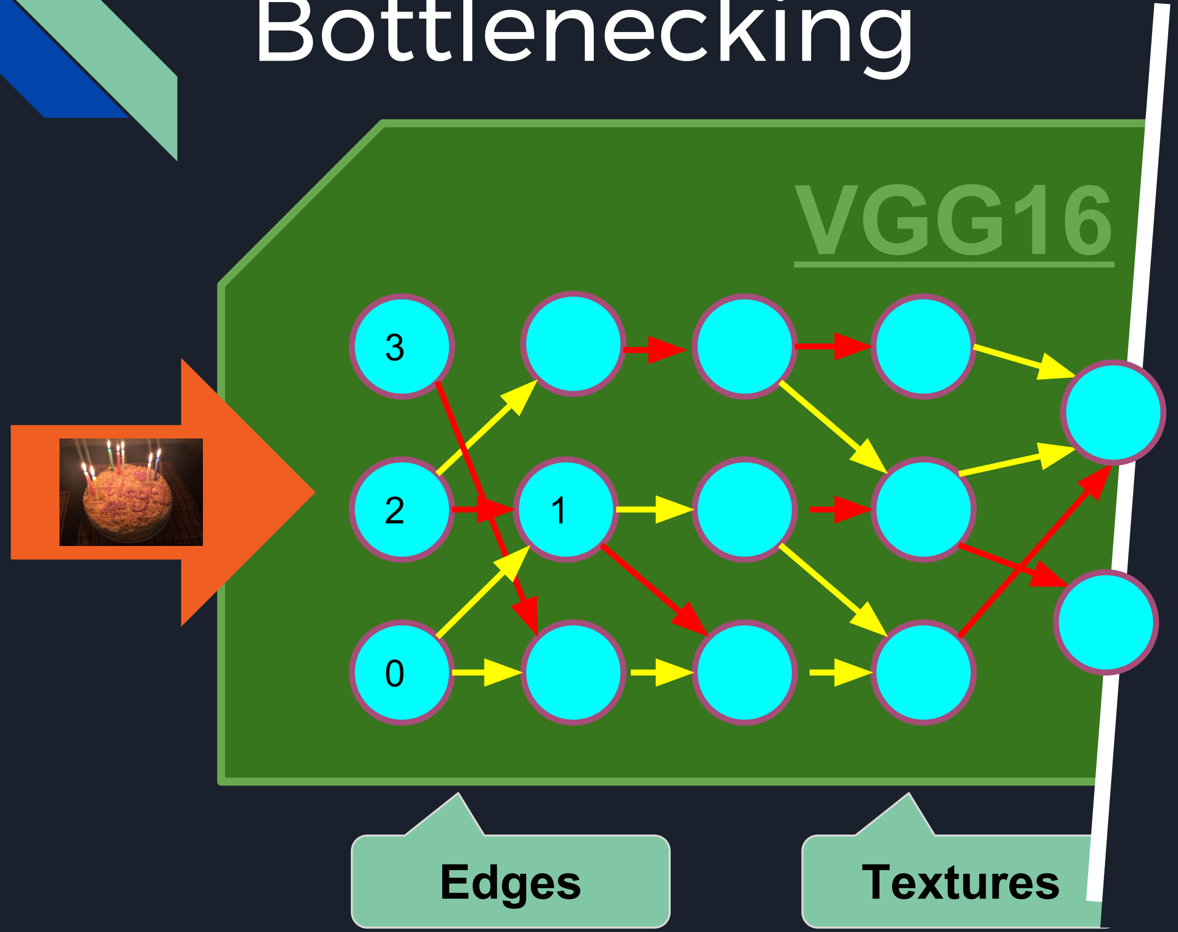
Bottlenecking



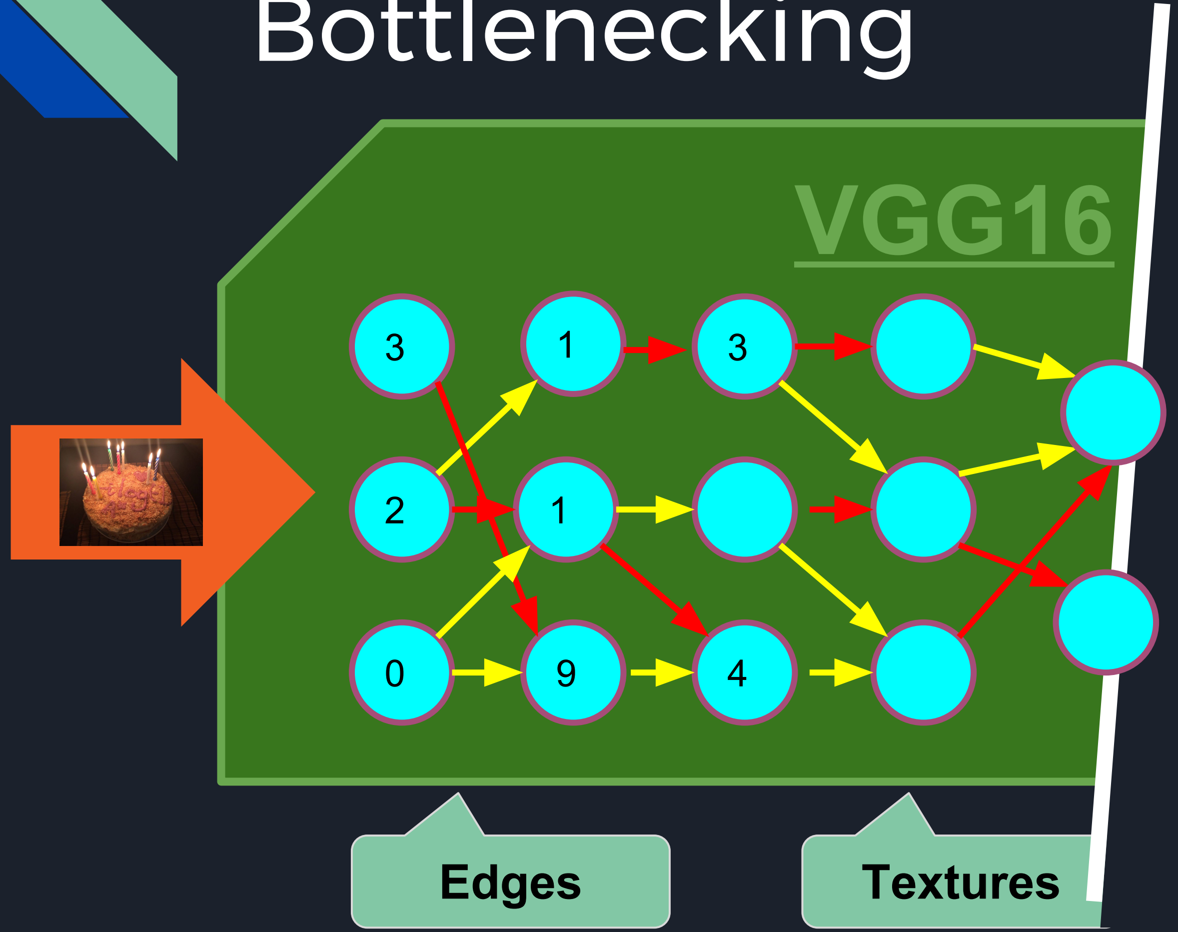
Bottlenecking



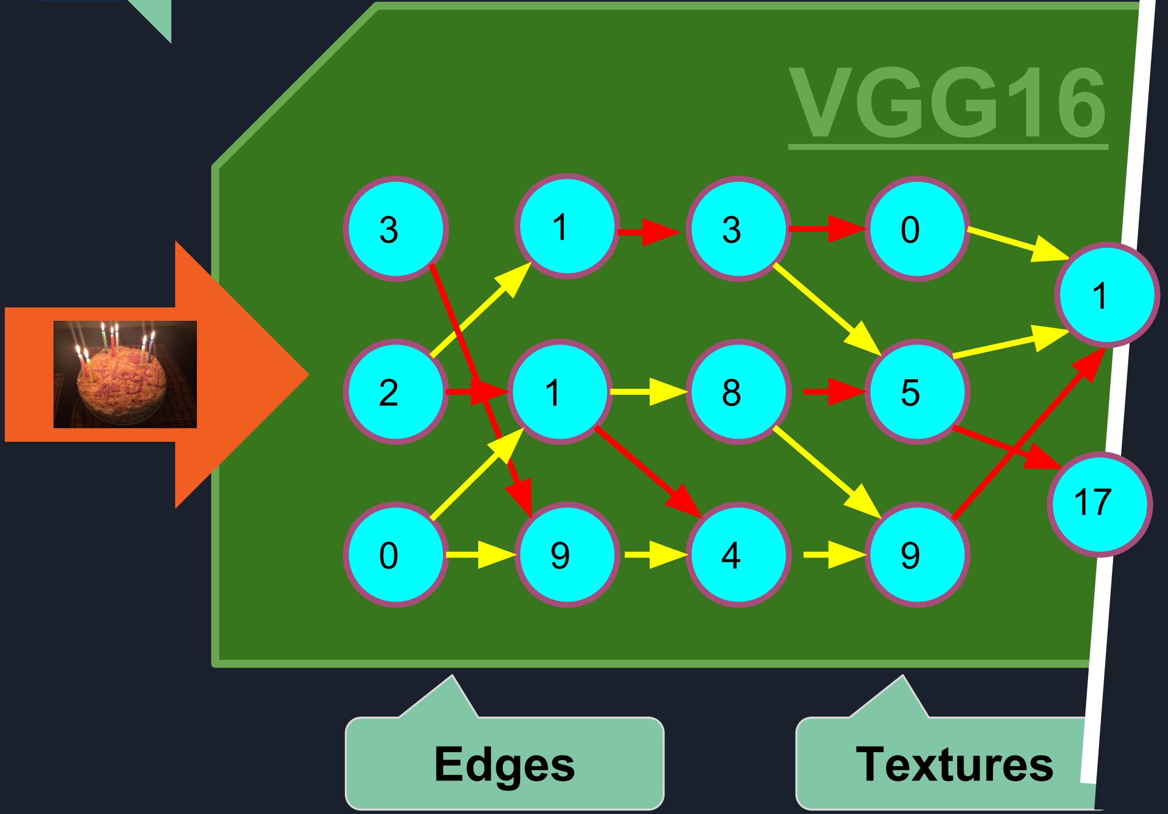
Bottlenecking



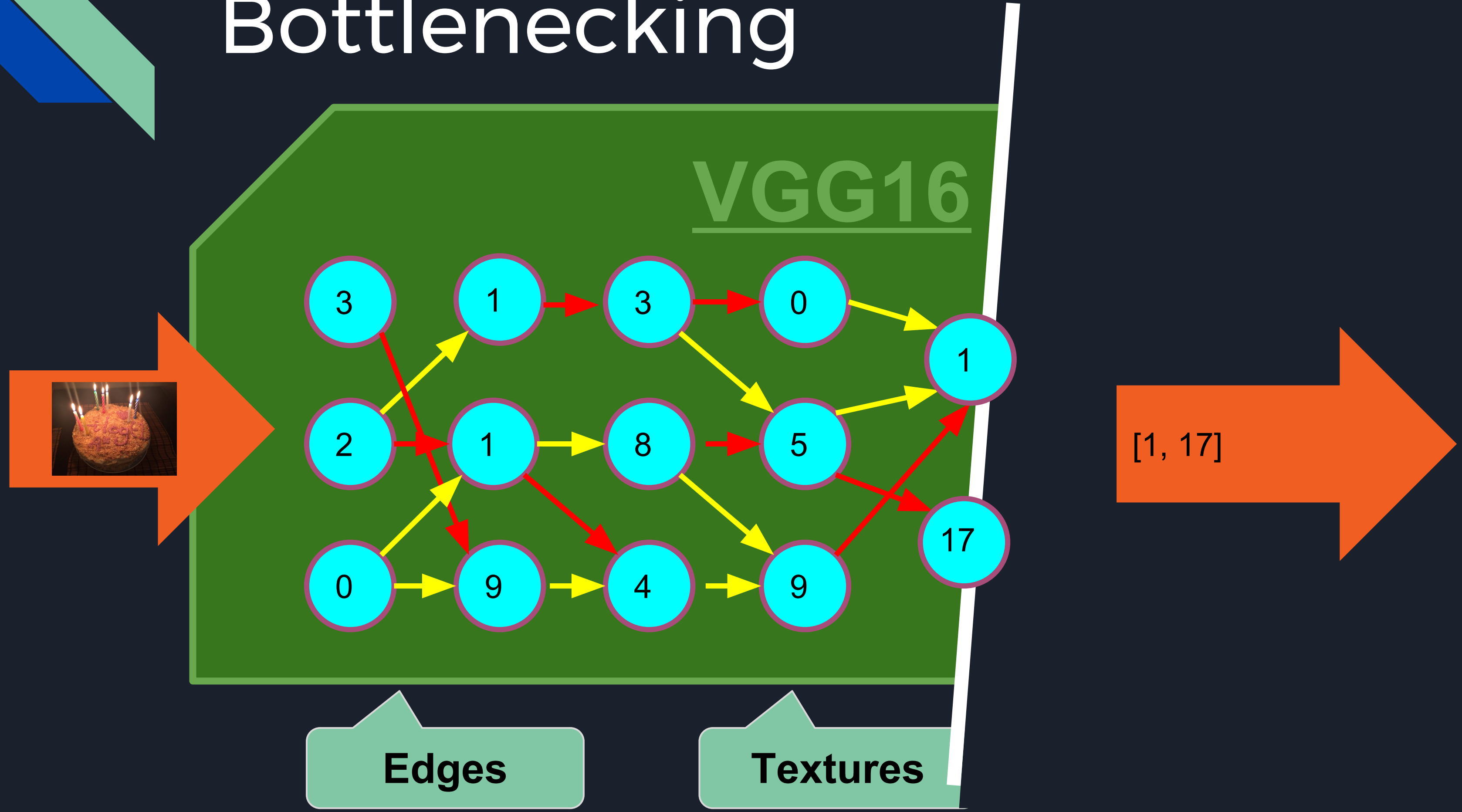
Bottlenecking



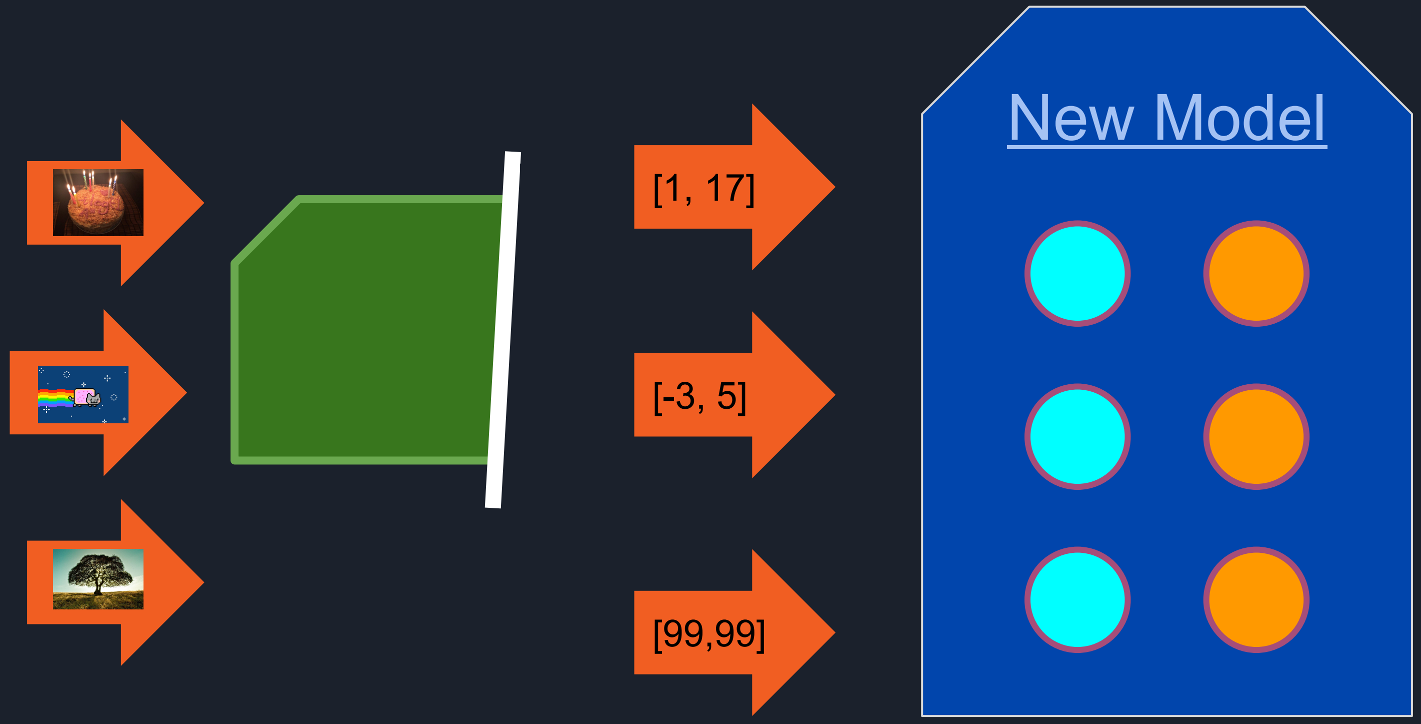
Bottlenecking



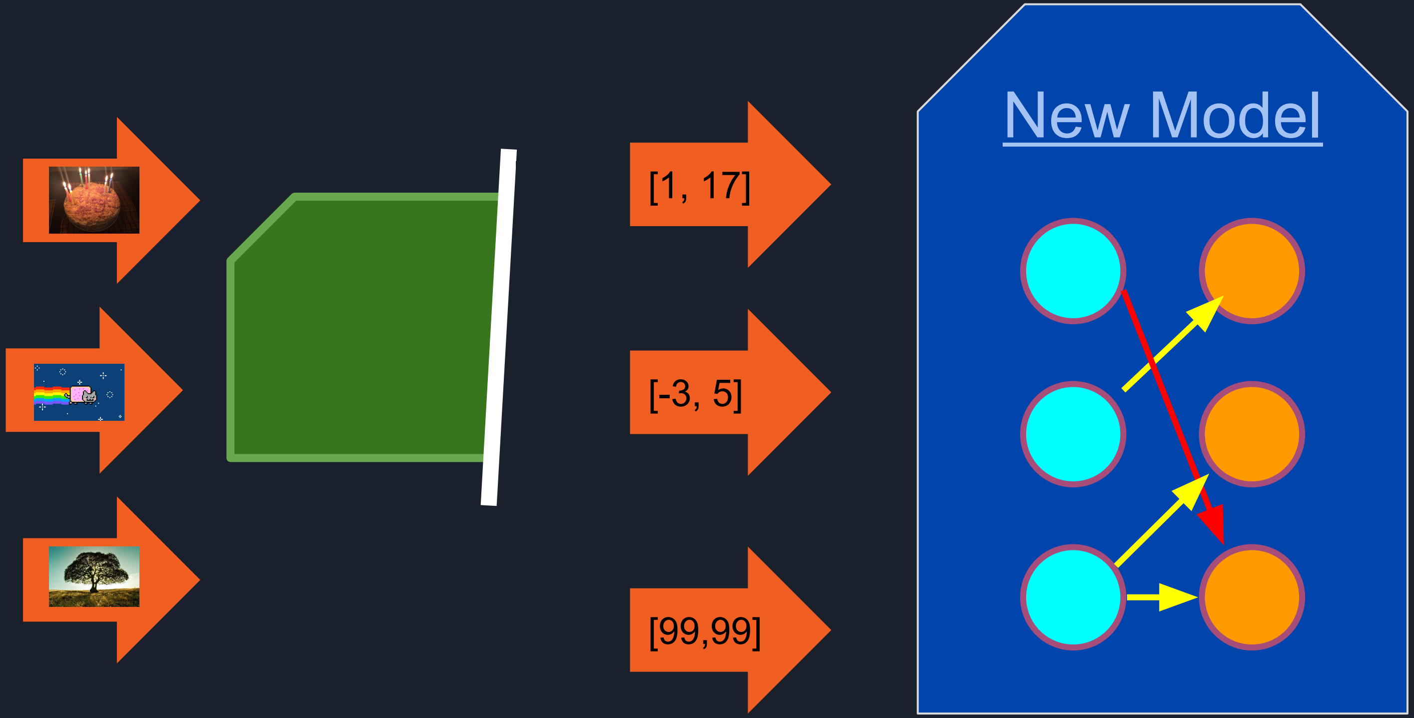
Bottlenecking



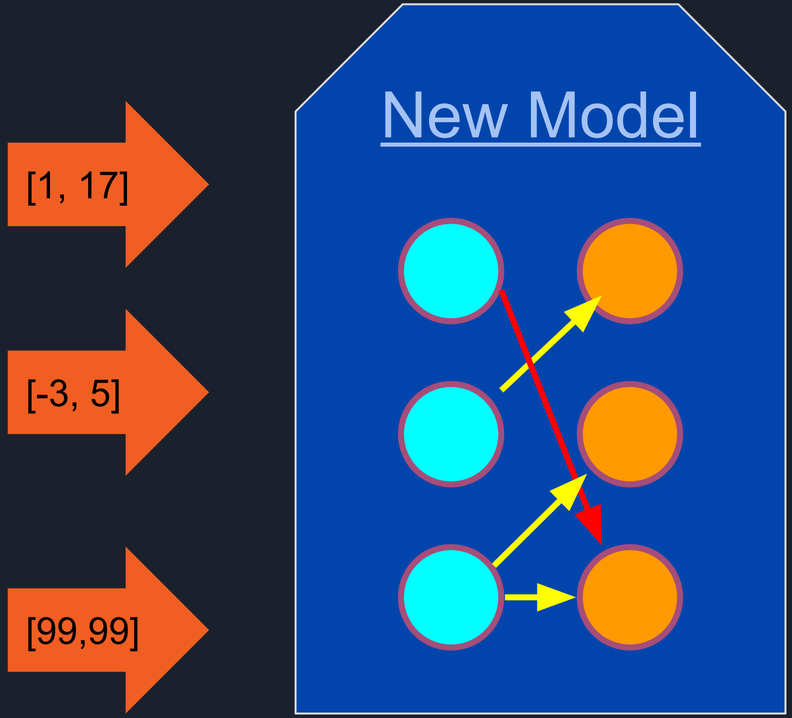
Bottlenecking



Bottlenecking



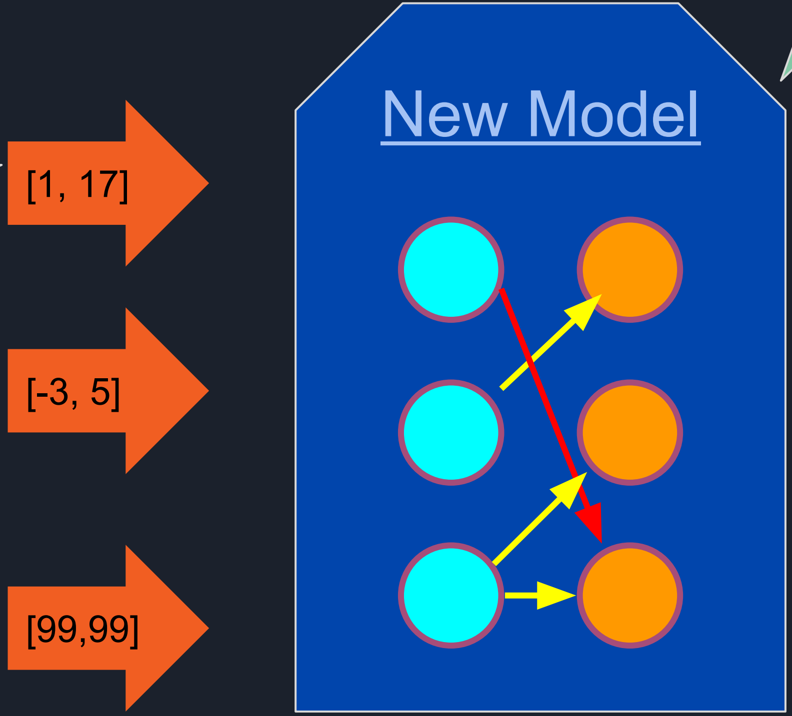
Bottlenecking



Bottlenecking

Model is smaller than VGG16 model

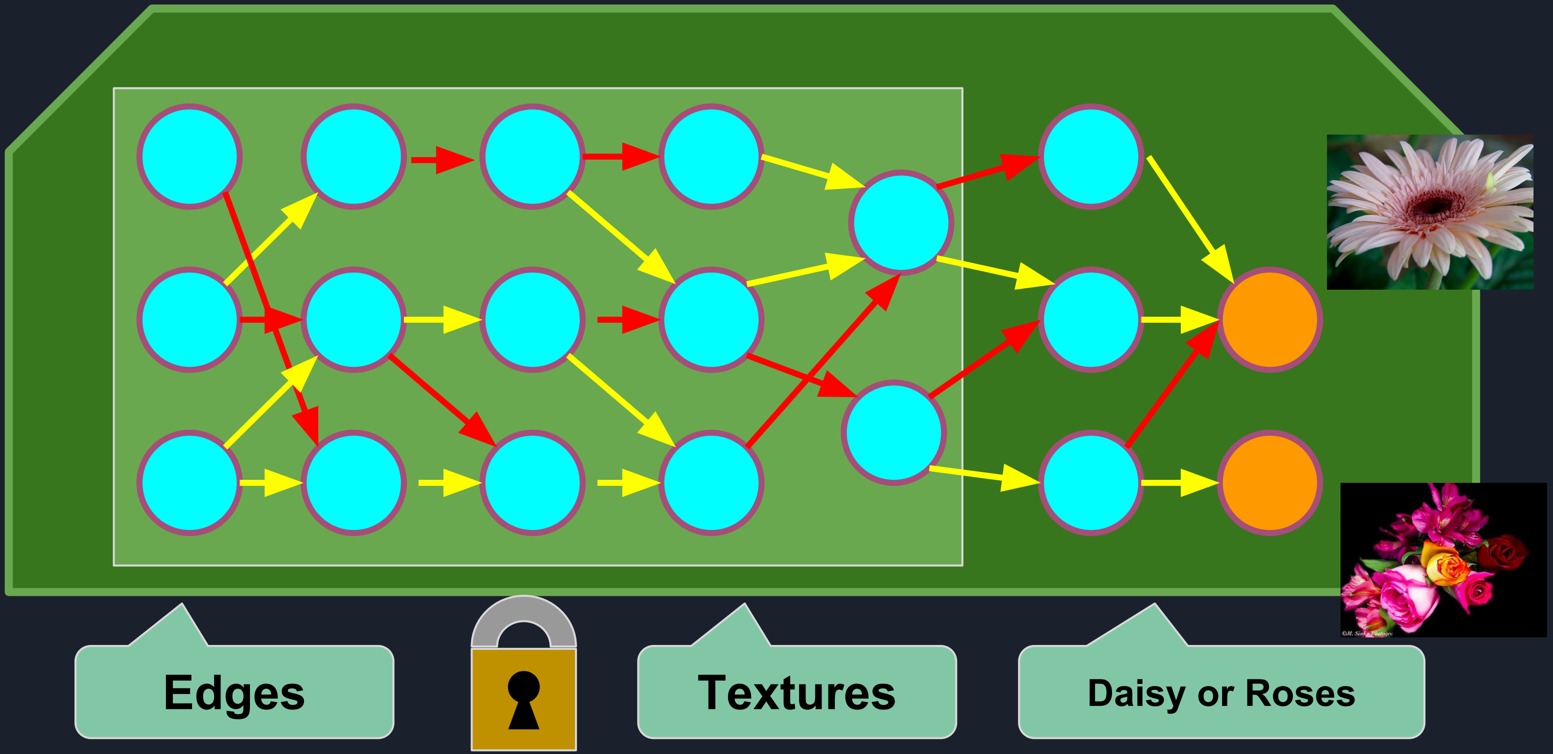
Weights are much smaller than image files



No longer need to save image files, or VGG16 model

Stock image set: Daisies Vs Roses

Transfer Learning (InceptionV3)



Transfer Learning (InceptionV3):

```
# create the base pre-trained model
base_model = InceptionV3(weights='imagenet', include_top=False)

# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x = Dense(1024, activation='relu')(x)

# and a logistic layer -- For 2 classes
predictions = Dense(2, activation='softmax')(x)

# this is the model we will train
model = Model(input=base_model.input, output=predictions)

# first: train only the top layers (which were randomly initialized)
# i.e. freeze all convolutional InceptionV3 layers
for layer in base_model.layers:
    layer.trainable = False

# compile the model (should be done *after* setting layers to non-trainable)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

Examples (InceptionV3):

[[0.02206078 0.97793919]]

[[0.9629941   0.03700593]]

Training time:
1-2 Hours

Accuracy on
Test set: ~94%

Results (Bottlenecking)

694 images
41Mb

5 minutes

VGG16

1	73	-5	0.6
76	5	-1	2.3
3	52	18	3.5
8	1.1	5.3	9.7

1 table
12.5Mb

Results (Bottlenecking)

1	73	-5	0.6
76	5	-1	2.3
3	52	18	3.5
8	1.1	5.3	9.7

New Model

2 dense layers
1 softmax layer

Training time: 12 seconds
Accuracy on Test set: 98%

Results (Comparison)

Given the tests on the flowers data set:

- Transfer Learning: inceptionV3 Training (1-2 hours)

- Bottlenecking: VGG16 Training (12 seconds)

We chose to do bottlenecking on the BCSA data

Results (w/ BCSA Data)

7100 images
~15 GB

60 minutes

VGG16

1	78	-5	0.6
7	5	3	2
3	5	1	3
8	1	5	9

1 Data table
27MB

Results (w/ BCSA Data)

1	73	-5	0.6
76	5	-1	2.3
3	52	18	3.5
8	1.1	5.3	9.7

New Model

2 dense layers
1 softmax layer

Training time:
73 seconds

Accuracy on Test set:
~79%

Results (w/ BCSA Data)

1	73	-5	0.6
76	5	-1	2.3
3	52	18	3.5
8	1.1	5.3	9.7

New Model

2 dense layers
1 softmax layer

Training time:
73 seconds

Accuracy on Test set:
~79%

BCSA Data:
High Hazard ~80%
Low Hazard ~20%

Caveats for Bottlenecking

- Bottlenecking only works if the locked nodes are well trained. If they make wrong predictions, then the end nodes will be trained incorrectly.
- Choosing the optimal amount of top layers to train.

Further Approaches

Rather than take objects from many categories and directly trying to determine their safety, determine safety for specific type of object first. (Hazards types vary between objects)

Design the network in two steps: 1) Determine what type of object is presented. 2) then determine its safety.

Acknowledgements

We would thank BCSA for the Data and the tutors:

(Doris and Soyean in particular)

As well as the BC Data workshop and its funders (and Aaron for organising)

And the various blogs we have stolen code from

References

- Keras (https://github.com/fchollet/keras)
- VGG16 (https://github.com/fastai/courses/blob/master/deeplearning1/nbs/lesson1.ipynb)
- “ Using Transfer Learning and Bottlenecking to Capitalize on State of the Art DNNs” (https://medium.com/@galen.ballew/transferlearning-b65772083b47)
- Flower Photos (http://download.tensorflow.org/example_im%E2%80%A6/flower_photos.tgz)