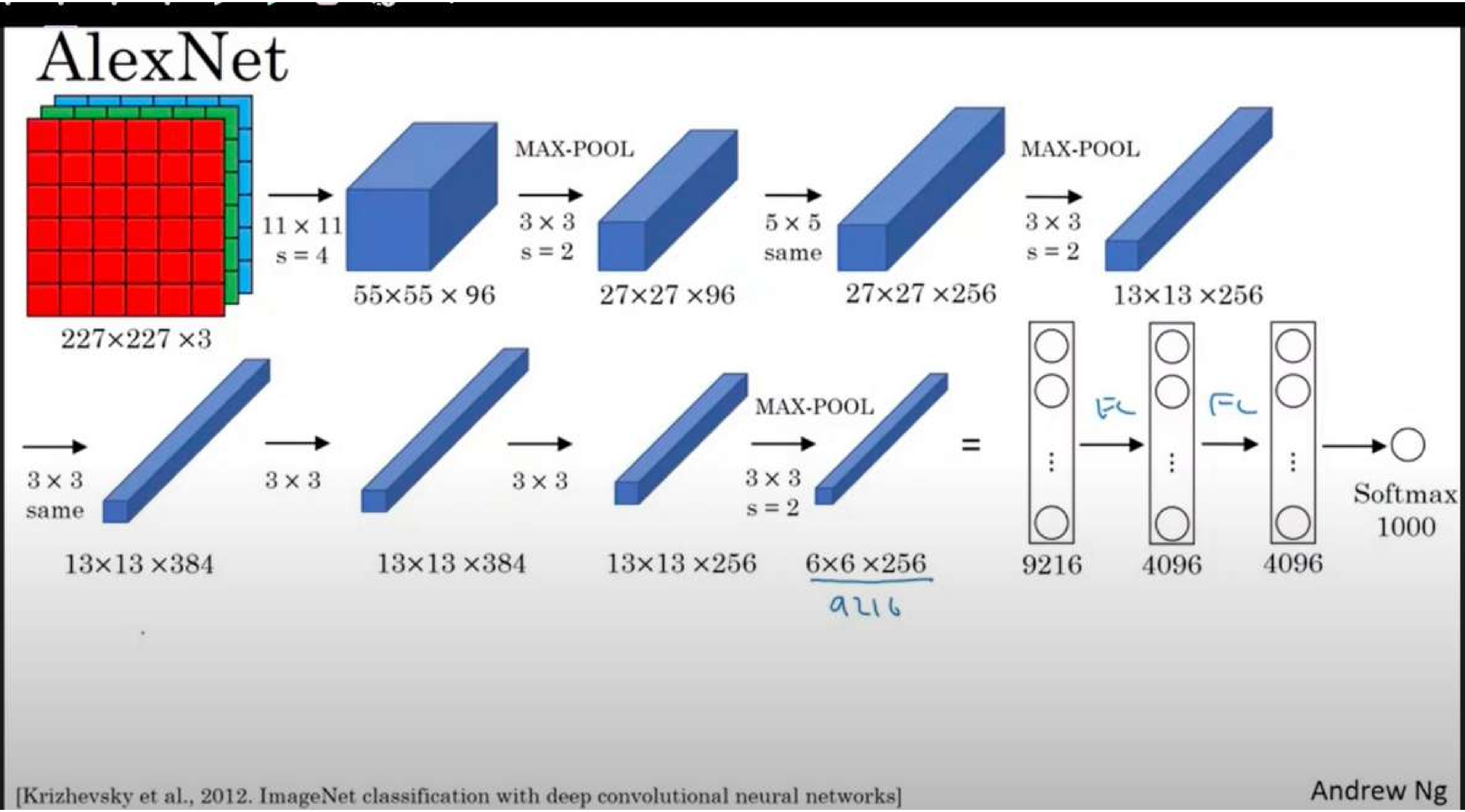


AlexNet

227×227 ×3

11 × 11
s = 4

55×55 × 96

MAX-POOL
3 × 3
s = 2

27×27 ×96

5 × 5
same

27×27 ×256

MAX-POOL
3 × 3
s = 2

13×13 ×256

3 × 3
same

13×13 ×384

3 × 3

13×13 ×384

3 × 3

13×13 ×256

MAX-POOL
3 × 3
s = 2

6×6 ×256

9216

=

9216

FC

4096

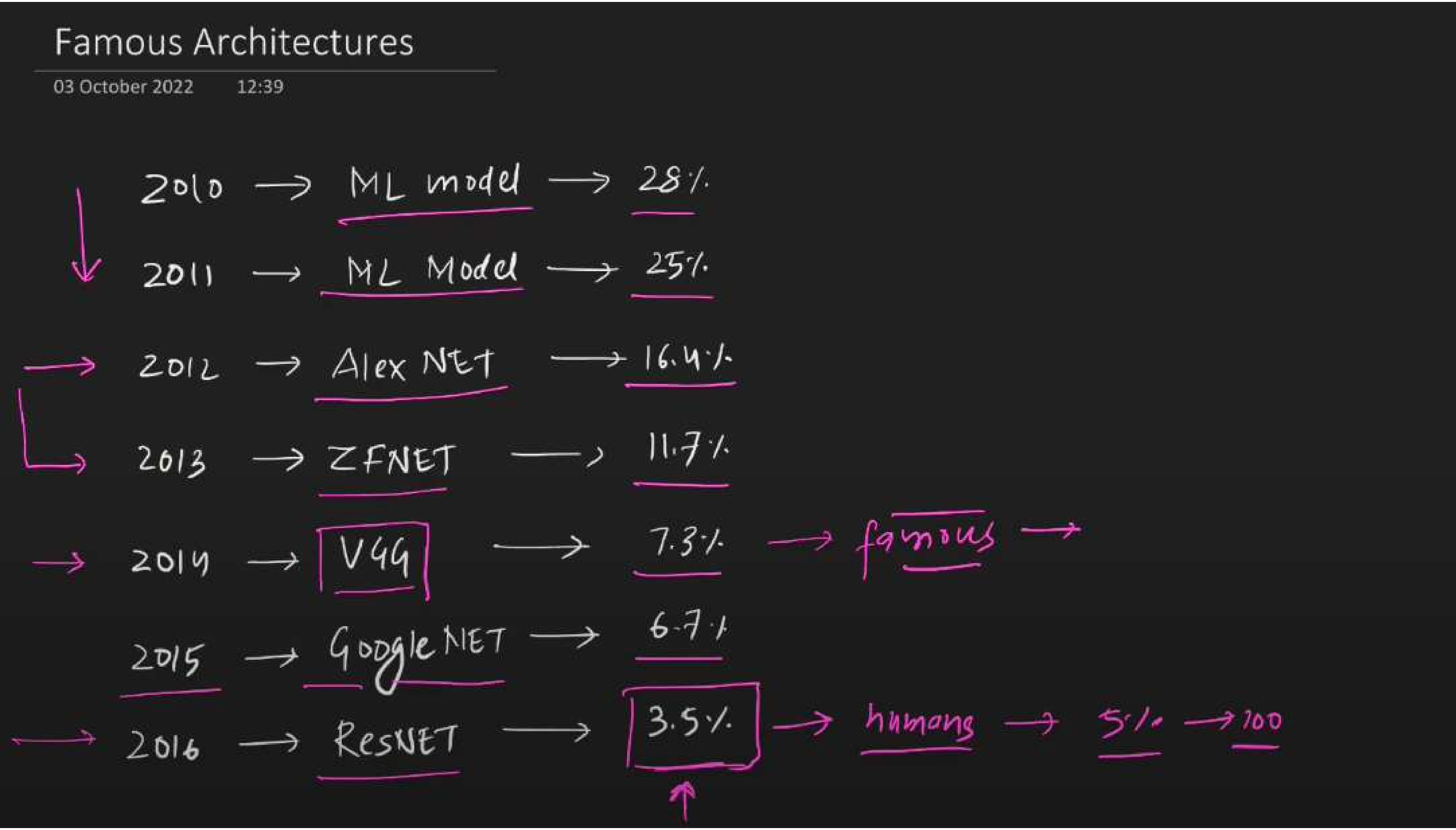
FC

4096

Softmax
1000

[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Andrew Ng



Famous Architectures

03 October 2022 12:39

2010 → ML model → 28%

2011 → ML Model → 25%

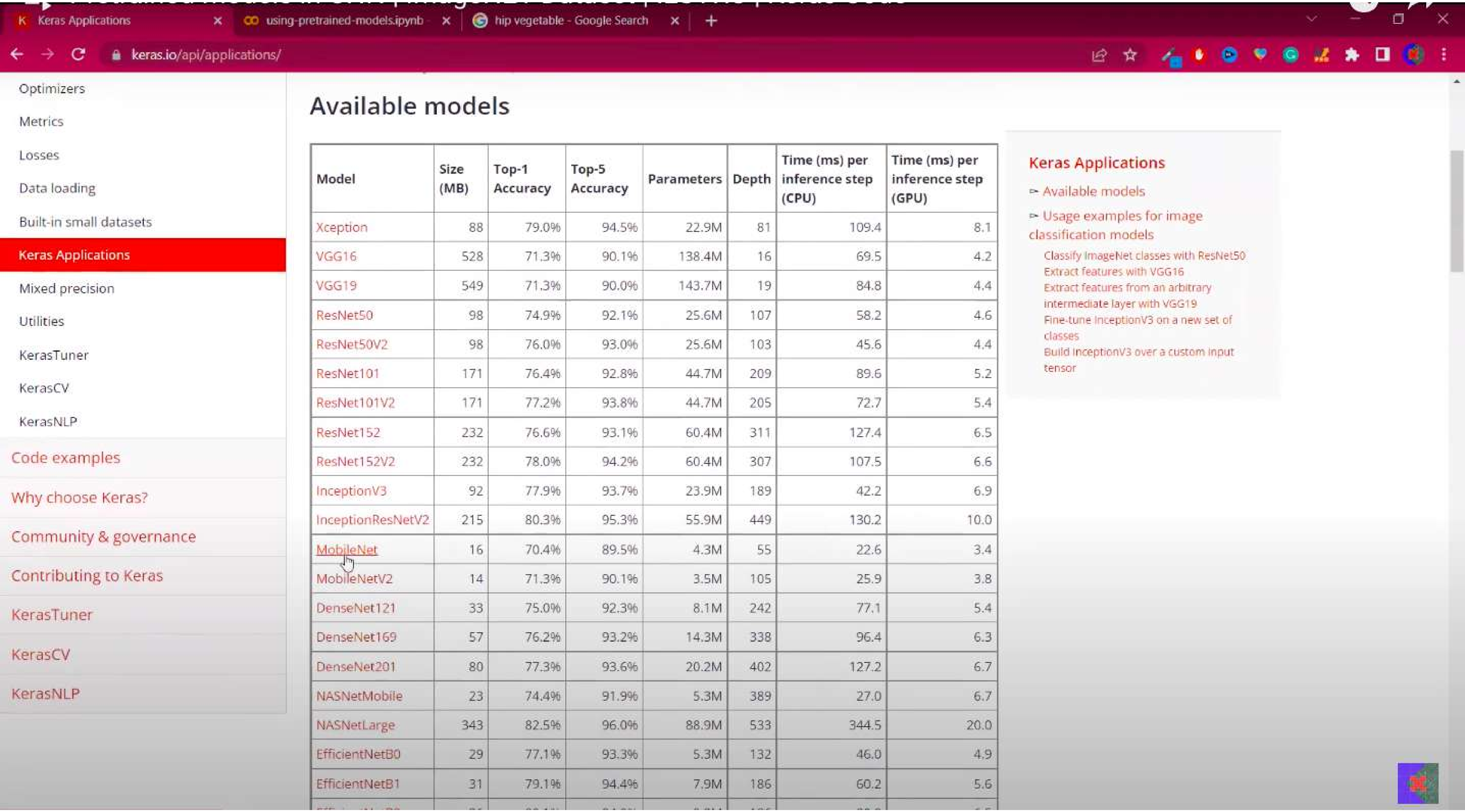
2012 → Alex NET → 16.4%

2013 → ZFNET → 11.7%

2014 → VGG → 7.3% → famous →

2015 → GoogleNET → 6.7%

2016 → ResNET → 3.5% → humans → 5% → 100

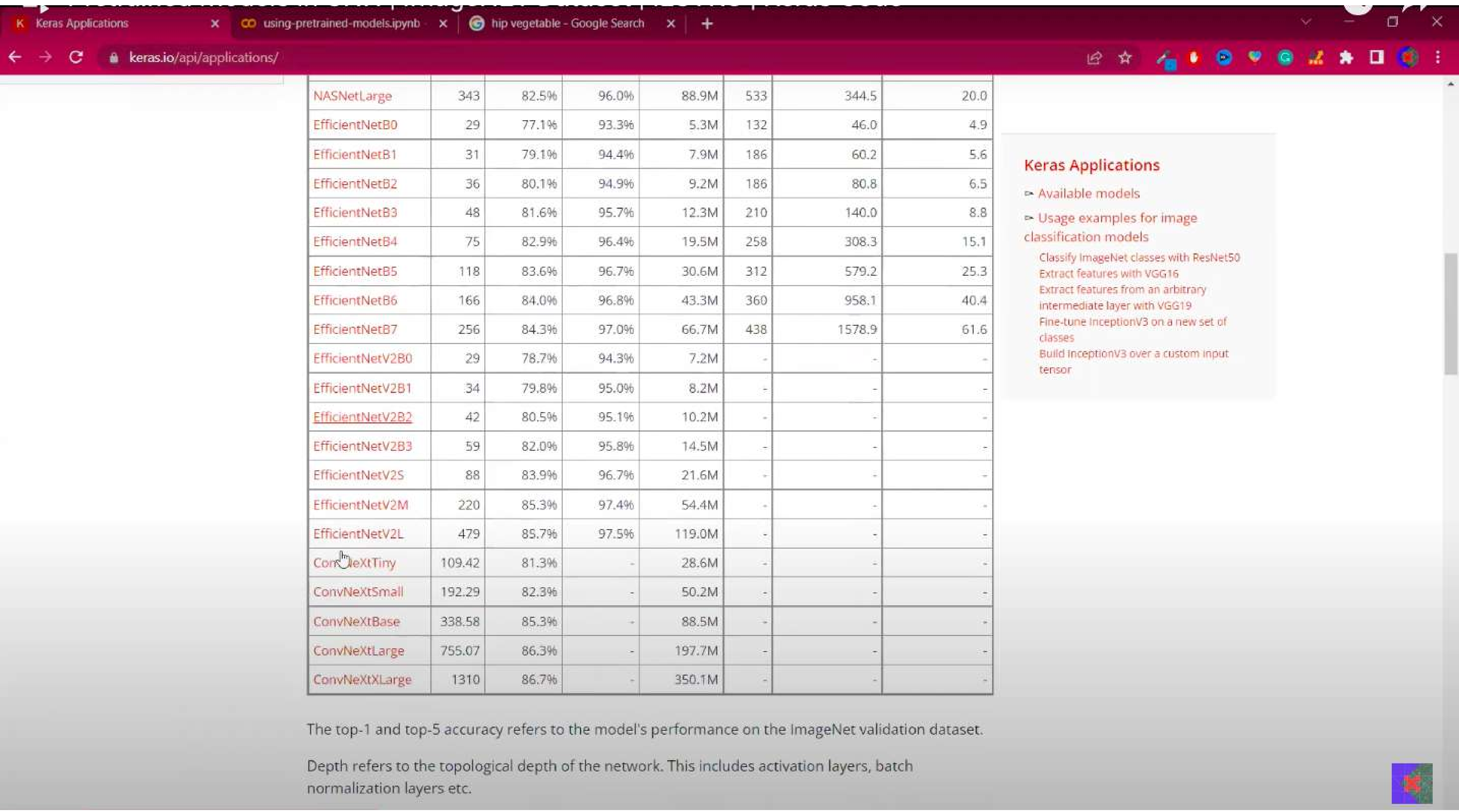


Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6

Keras Applications

- Available models
- Usage examples for image classification models
  - Classify ImageNet classes with ResNet50
  - Extract features with VGG16
  - Extract features from an arbitrary intermediate layer with VGG19
  - Fine-tune InceptionV3 on a new set of classes
  - Build InceptionV3 over a custom input tensor


Keras Applications
using-pretrained-models.ipynb
hip vegetable - Google Search
keras.io/api/applications/

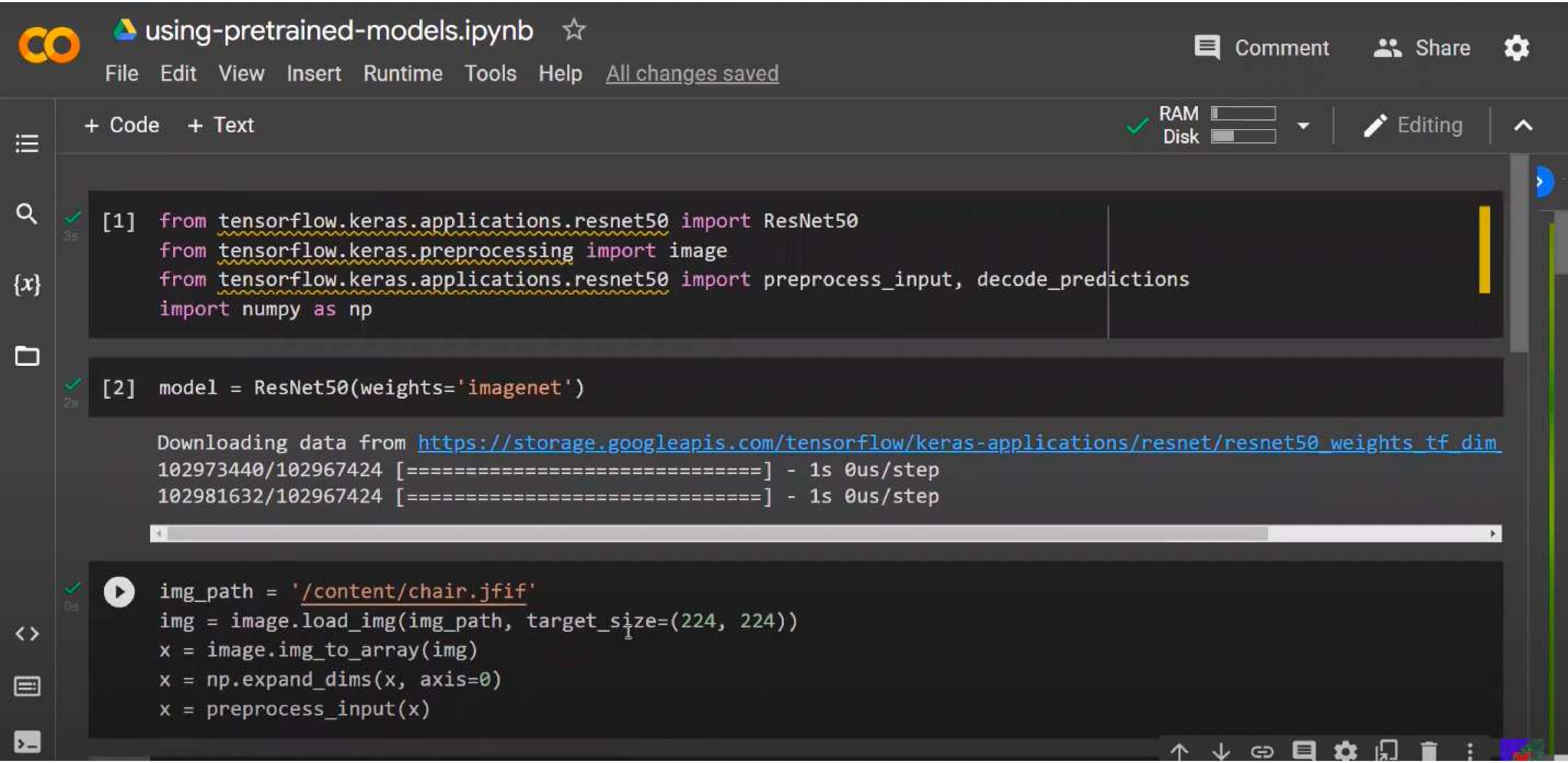
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5
EfficientNetB3	48	81.6%	95.7%	12.3M	210	140.0	8.8
EfficientNetB4	75	82.9%	96.4%	19.5M	258	308.3	15.1
EfficientNetB5	118	83.6%	96.7%	30.6M	312	579.2	25.3
EfficientNetB6	166	84.0%	96.8%	43.3M	360	958.1	40.4
EfficientNetB7	256	84.3%	97.0%	66.7M	438	1578.9	61.6
EfficientNetV2B0	29	78.7%	94.3%	7.2M	-	-	-
EfficientNetV2B1	34	79.8%	95.0%	8.2M	-	-	-
EfficientNetV2B2	42	80.5%	95.1%	10.2M	-	-	-
EfficientNetV2B3	59	82.0%	95.8%	14.5M	-	-	-
EfficientNetV2S	88	83.9%	96.7%	21.6M	-	-	-
EfficientNetV2M	220	85.3%	97.4%	54.4M	-	-	-
EfficientNetV2L	479	85.7%	97.5%	119.0M	-	-	-
ConvNeXtTiny	109.42	81.3%	-	28.6M	-	-	-
ConvNeXtSmall	192.29	82.3%	-	50.2M	-	-	-
ConvNeXtBase	338.58	85.3%	-	88.5M	-	-	-
ConvNeXtLarge	755.07	86.3%	-	197.7M	-	-	-
ConvNeXtXLarge	1310	86.7%	-	350.1M	-	-	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

Depth refers to the topological depth of the network. This includes activation layers, batch normalization layers etc.

Keras Applications

- Available models
- Usage examples for image classification models
  - Classify ImageNet classes with ResNet50
  - Extract features with VGG16
  - Extract features from an arbitrary intermediate layer with VGG19
  - Fine-tune InceptionV3 on a new set of classes
  - Build InceptionV3 over a custom input tensor

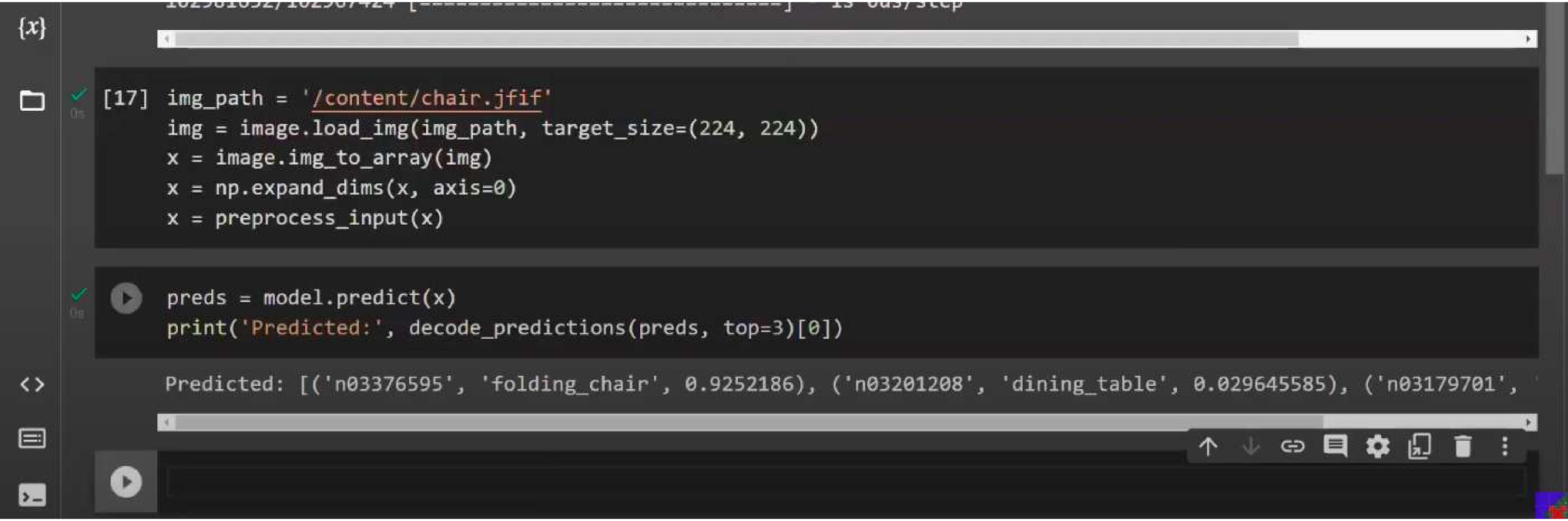

using-pretrained-models.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM
Disk
Editing

```
[1] from tensorflow.keras.applications.resnet50 import ResNet50
    from tensorflow.keras.preprocessing import image
    from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
    import numpy as np
```

```
[2] model = ResNet50(weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim
102973440/102967424 [==============================] - 1s 0us/step
102981632/102967424 [==============================] - 1s 0us/step
```

```
img_path = '/content/chair.jfif'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
```

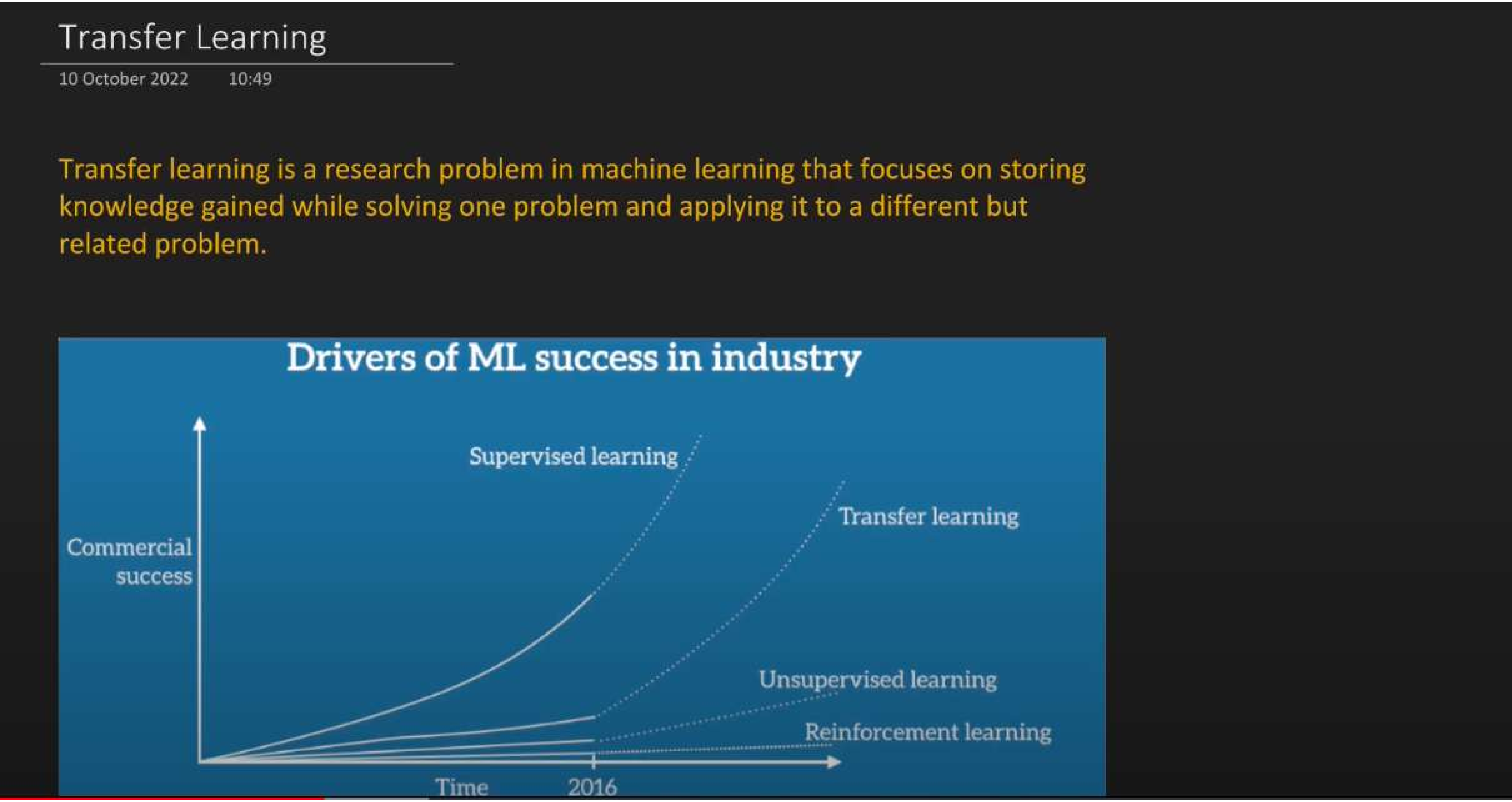


```
102981632/102967424 [==============================] - 1s 0us/step
```

```
[17] img_path = '/content/chair.jfif'
     img = image.load_img(img_path, target_size=(224, 224))
     x = image.img_to_array(img)
     x = np.expand_dims(x, axis=0)
     x = preprocess_input(x)
```

```
preds = model.predict(x)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

```
Predicted: [('n03376595', 'folding_chair', 0.9252186), ('n03201208', 'dining_table', 0.029645585), ('n03179701',
```

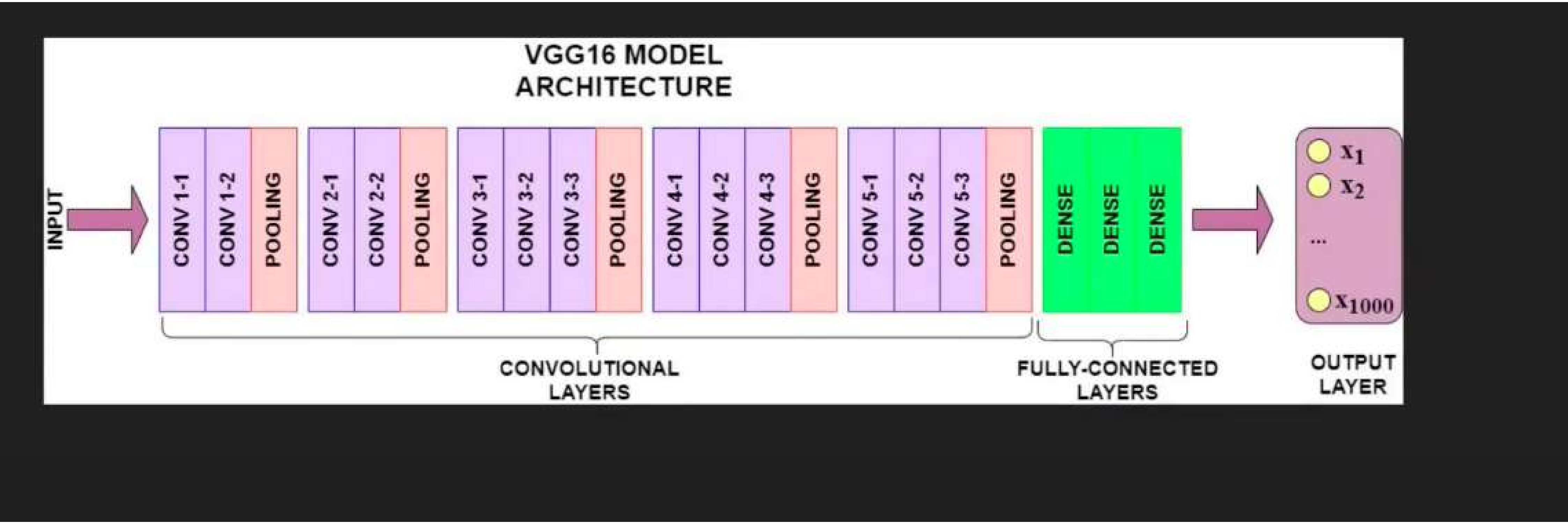


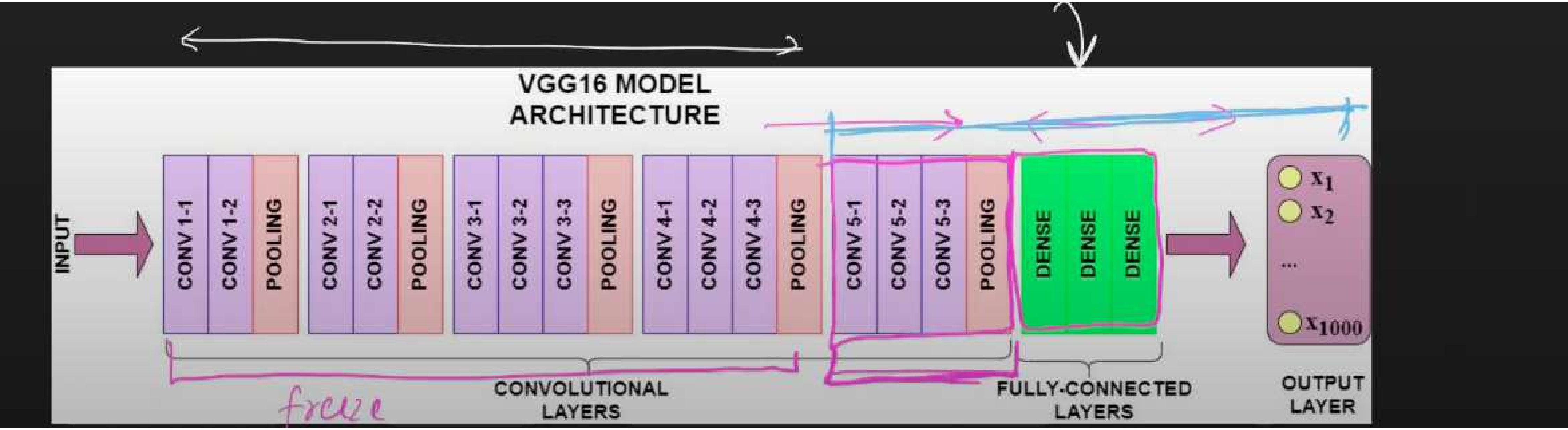
Transfer Learning

10 October 2022 10:49

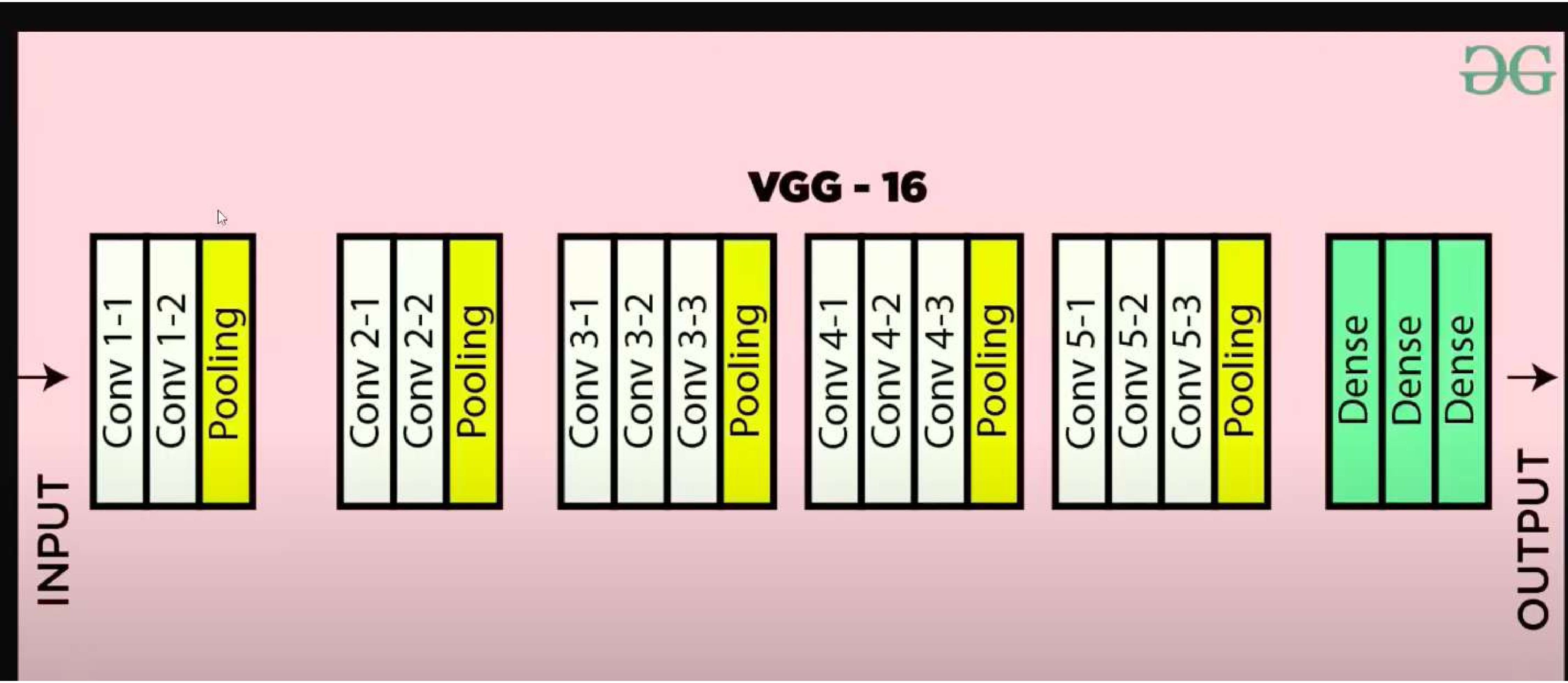
Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

Drivers of ML success in industry


VGG16 MODEL
ARCHITECTURE
INPUT
CONV 1-1
CONV 1-2
POOLING
CONV 2-1
CONV 2-2
POOLING
CONV 3-1
CONV 3-2
CONV 3-3
POOLING
CONV 4-1
CONV 4-2
CONV 4-3
POOLING
CONV 5-1
CONV 5-2
CONV 5-3
POOLING
DENSE
DENSE
DENSE
x_1
x_2
...
x_{1000}
CONVOLUTIONAL
LAYERS
FULLY-CONNECTED
LAYERS
OUTPUT
LAYER



VGG16 MODEL ARCHITECTURE



VGG - 16

INPUT

Conv 1-1

Conv 1-2

Pooling

Conv 2-1

Conv 2-2

Pooling

Conv 3-1

Conv 3-2

Conv 3-3

Pooling

Conv 4-1

Conv 4-2

Conv 4-3

Pooling

Conv 5-1

Conv 5-2

Conv 5-3

Pooling

Dense

Dense

Dense

OUTPUT