

ML for SS: Neural Networks for Image Classification

Dr. Richard M. Crowley

rcrowley@smu.edu.sg

<https://rmc.link/>



Overview

Papers

Liu, Dzyabura and Mizik (2020)

- Examines brand image and how reflective profiles are of the brands

Zhang, Lee, Singh and Srinivasan (2017)

- Examines how images in listings impact AirBNB properties

Aubry, Kraeusl, Manso, and Spaenjers (2022)

- Estimation errors in art auction listings

Technical Discussion: DoubleML

Focus on Neural Networks for images

Python

- Using Keras with Tensorflow for image classification
 1. Repeat our MNIST example using a proper CNN
 2. Using a premade GAN approach for even higher performance
- Using a 80-class pretrained classifier
- Combining images and text with CLIP

R

- You can use Keras from R through RStudio's package

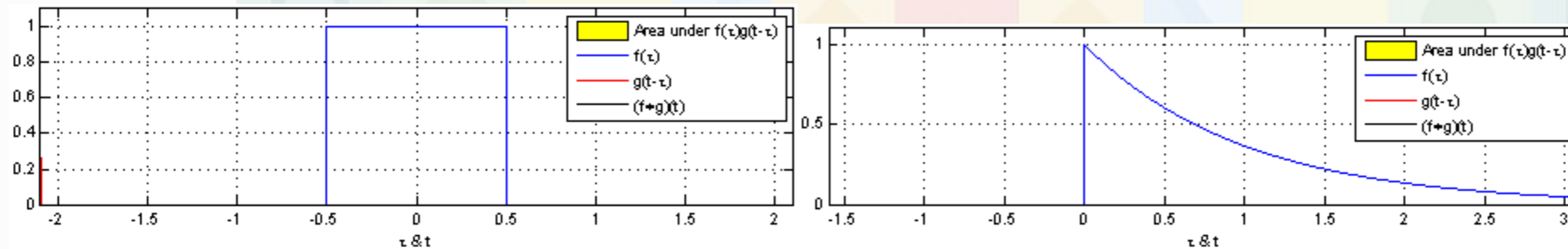
Python's support is a lot better here



MNIST: Extending to a CNN

How CNNs work

- CNNs use repeated convolution, usually looking at slightly bigger chunks of data each iteration
- But what is convolution? It is illustrated by the following graphs (from [Wikipedia](#)):



Further reading

Setup

- The setup is similar, except we don't need to reshape our X data
- We do need to add an additional dimension to our images though, which `np.expand_dims()` does for us

```
(train_X, train_Y), (test_X, test_Y) = keras.datasets.mnist.load_data()

train_X = train_X.astype("float32") / 255
test_X = test_X.astype("float32") / 255

train_X = np.expand_dims(train_X, -1)
test_X = np.expand_dims(test_X, -1)

train_Y = keras.utils.to_categorical(train_Y, 10)
test_Y = keras.utils.to_categorical(test_Y, 10)

print('Train, X:%s, Y:%s' % (train_X.shape, train_Y.shape))
print('Test, X:%s, Y:%s' % (test_X.shape, test_Y.shape))
```

```
Train, X:(60000, 28, 28, 1), Y:(60000, 10)
Test, X:(10000, 28, 28, 1), Y:(10000, 10)
```

Build the model

- Here we use `Conv2D()` layers for the convolution
- The `MaxPooling2D()` layers downsample (shrink) the data
- The `Flatten()` layer reshapes the output to a vector
- `Relu` is essentially the same as a call option payoff (“hockey stick”)
- `Softmax` is to output the class with the highest weight (argmax)

```
num_classes = 10
input_shape = (28, 28, 1)
model_cnn = keras.Sequential(
    [
        keras.layers.InputLayer(input_shape=input_shape),
        keras.layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        keras.layers.MaxPooling2D(pool_size=(2, 2)),
        keras.layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        keras.layers.MaxPooling2D(pool_size=(2, 2)),
        keras.layers.Flatten(),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(num_classes, activation="softmax"),
    ]
)
model_cnn.summary()
```


Build the model

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dropout_2 (Dropout)	(None, 1600)	0

Fit the model and evaluate

- Fitting and evaluating is the same as before

```
batch_size = 128
epochs = 10

model_cnn.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
model_cnn.fit(train_X, train_Y, batch_size=batch_size, epochs=epochs, validation_split=0.1)

score = model_cnn.evaluate(test_X, test_Y, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

Test loss: 0.0291274506598711

Test accuracy: 0.9897000193595886

What does the model get right?



What does the model get wrong?





Explaining a CNN

SHAP and TensorFlow

- Recall that Wich, Bauer and Groh (2020 WOAH) used `shap.DeepExplainer()` to analyze a neural network
 - We can do the same!
- First, feed SHAP the model and some sample images

```
images = np.random.randint(0, train_X.shape[0], size=25)  
e = shap.DeepExplainer(model_cnn, train_X[images])
```

- Then we will select 1 of each digit that the CNN got correct and incorrect

```
correct = [np.where((np.argmax(model_cnn.predict(test_X), axis=-1) == np.argmax(test_Y, axis=-1)) & \  
                  (np.argmax(test_Y, axis=-1) == i))[0][0] for i in range(0, 10)]  
incorrect = [np.where((np.argmax(model_cnn.predict(test_X), axis=-1) != np.argmax(test_Y, axis=-1)) & \  
                    (np.argmax(test_Y, axis=-1) == i))[0][0] for i in range(0, 10)]
```

SHAP for correct images



```
shap_values = e.shap_values(test_X[correct])  
shap.image_plot(shap_values, -test_X[correct])
```



SHAP for incorrect images

```
shap_values = e.shap_values(test_X[incorrect])  
shap.image_plot(shap_values, -test_X[incorrect])
```



Recent attempts at explaining CNNs

- Google & Stanford's "Automated Concept-based Explanation"

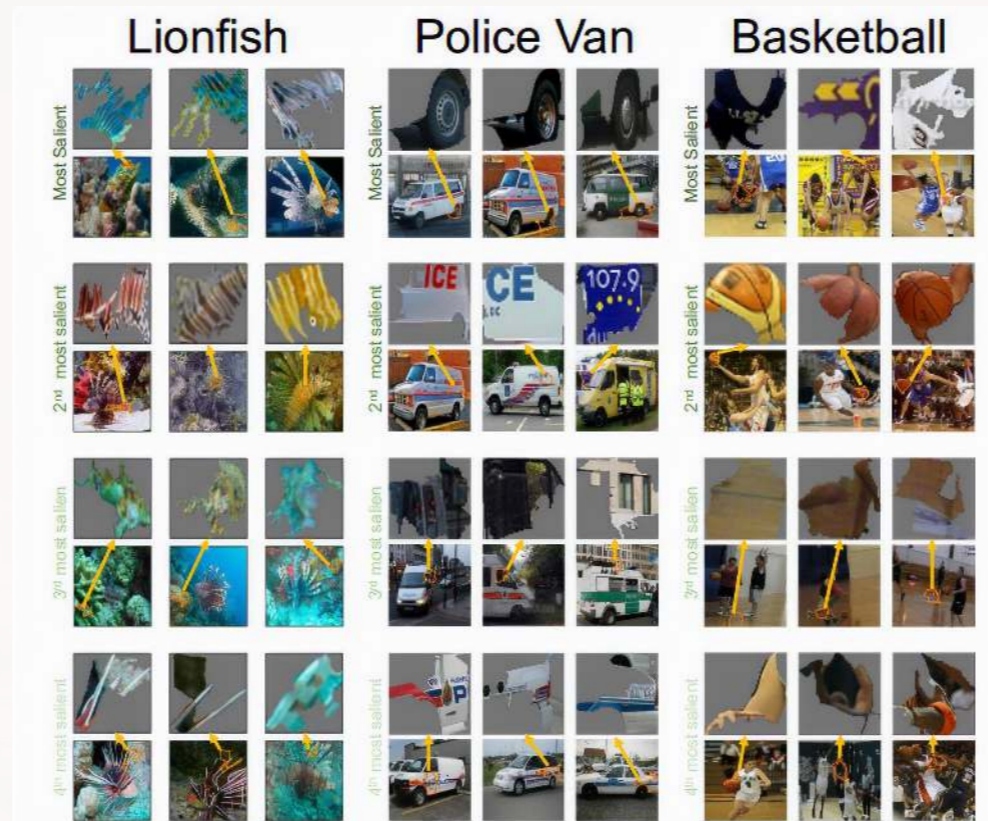


Figure 2: The output of ACE for three ImageNet classes. Here we depict three randomly selected examples of the top-4 important concepts of each class (each example is shown above the original image it was segmented from). Using this result, for instance, we could see that the network classifies police vans using the van's tire and the police logo.



Working with pretrained models

Where can I find pretrained models?

- There are many pretrained models on [TensorFlow Hub](#)
- There are also models contained in the TensorFlow Github page:
 - [Research models](#)
 - [Community models](#)
- Google Brain also maintains a collection of models in [trax](#)

Other platforms also maintain model collections

- PyTorch has [PyTorch Hub](#)
- Hugging Face maintains a [large collection of text models](#)
- ONNX maintains a collection of [framework-agnostic models](#)

We will look at TensorFlow Hub today

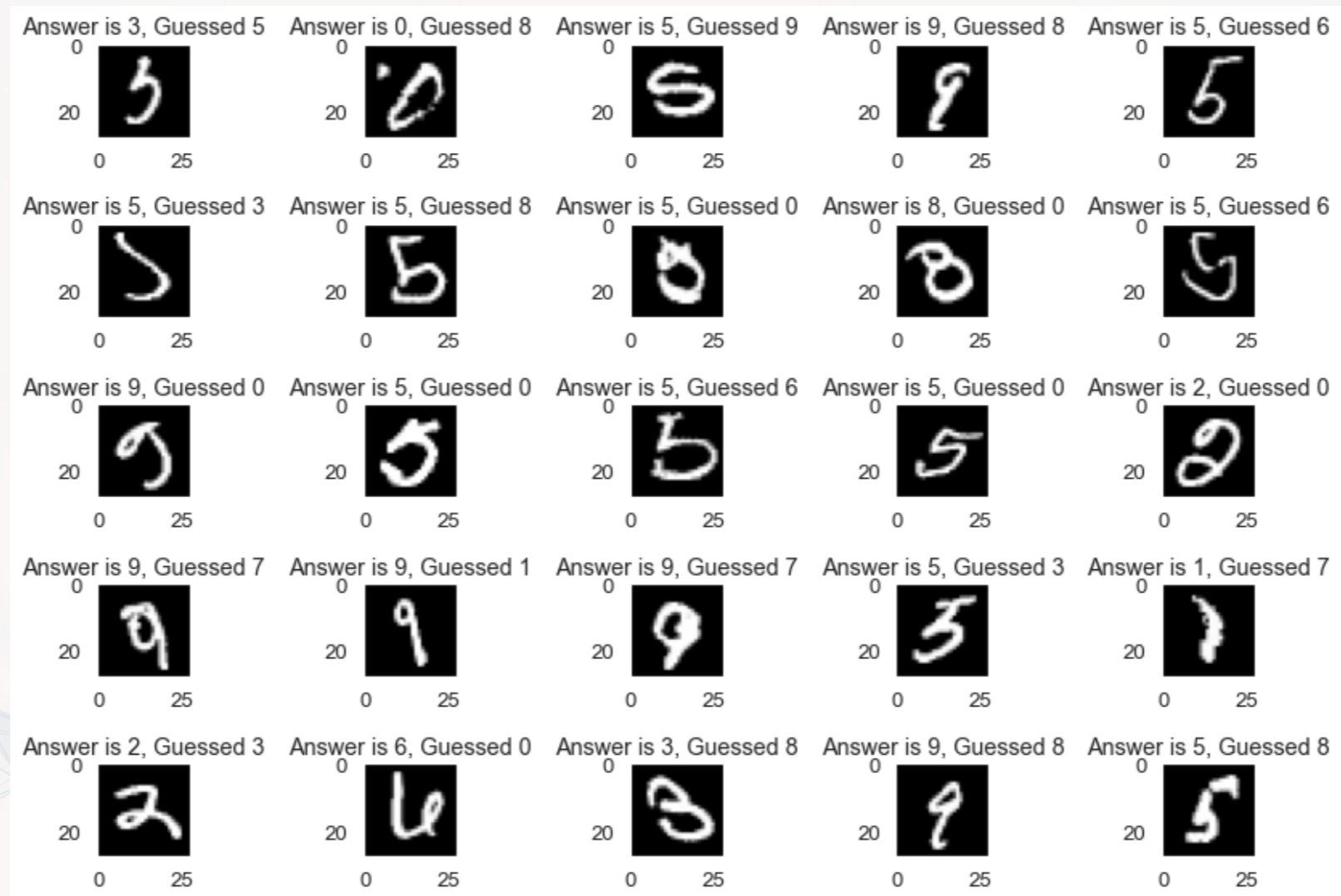
MNIST off-the-shelf

- The model we will be using is GAN-based MNIST classifier
 - [tfgan/eval/mnist/logits](https://tfhub.dev/tensorflow/tfgan/eval/mnist/logits)
- Use `hub.load()` to load in a model
- Apply it to our testing data, same as before
 - Just apply the model to our data

```
model_tfgan = hub.load("https://tfhub.dev/tensorflow/tfgan/eval/mnist/logits/1")
logits = model_tfgan(test_X).numpy()

# Check accuracy
sum(np.argmax(logits, -1) == np.argmax(test_Y, -1))
```

Examine incorrect answers





Object detection off-the-shelf

COCO Classification problem

- There are a lot of options for this
- We will use a model trained on [COCO](#) from CenterNet
 - [centernet/hourglass_512x512](#)
- This can detect 80 different object types, including people



```
# Full list of object types
labels = load_COCO_labelmap()
print(list(labels.values()))
```

```
['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train',
'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter',
'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear',
'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase',
'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat',
'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle',
'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake',
'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv',
'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven',
'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors',
'teddy bear', 'hair drier', 'toothbrush']
```

What is Centernet/Hourglass?

- Centernet is an approach that's intended to be used for drawing bounding boxes around objects
 - From Zhou, Wang, and Krähenbühl (2019)
 - The second stage in the classification problem for computer vision:
 1. Detect objects
 2. Locate them in the image
- Hourglass is a neural network structure based on CNNs

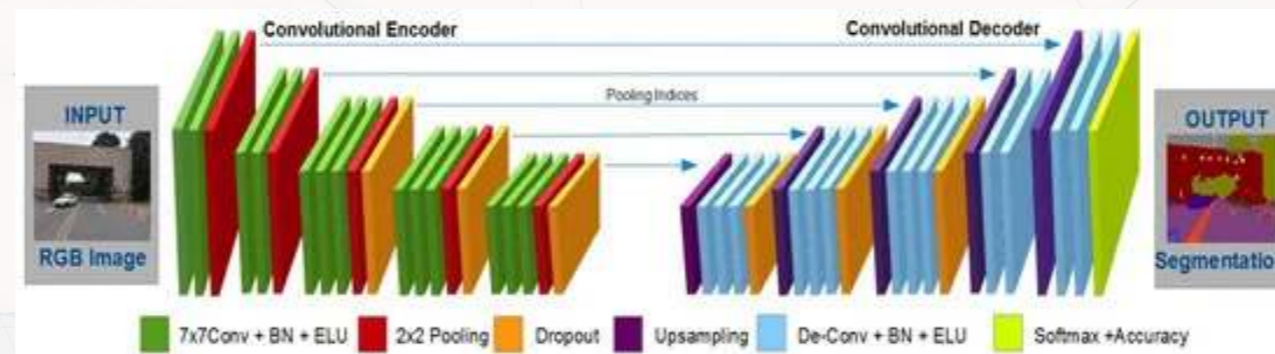


Image is Yasrab, Gu, and Zhang (2017), Figure 2

Using the model

```
centernet = hub.load('https://tfhub.dev/tensorflow/centernet/hourglass_512x512/1')  
  
image1, image1_np = load_image('../Data/S12_1.jpeg')  
image2, image2_np = load_image('https://pbs.twimg.com/media/E8ZIIKGXIAAipIh?format=jpg&name=small')
```



```
-rw-r--r-- 1 root root 9916885161 Aug 9 15:13 RC_2021-04-30  
-rw-r--r-- 1 root root 9223343241 Aug 9 15:30 RC_2021-05-01  
-rw-r--r-- 1 root root 9646977002 Aug 9 15:48 RC_2021-05-02  
-rw-r--r-- 1 root root 9790222766 Aug 9 16:06 RC_2021-05-03  
-rw-r--r-- 1 root root 9629653589 Aug 9 16:23 RC_2021-05-04  
-rw-r--r-- 1 root root 10128104379 Aug 9 16:43 RC_2021-05-05  
-rw-r--r-- 1 root root 10030968634 Aug 9 17:06 RC_2021-05-06  
-rw-r--r-- 1 root root 9640296547 Aug 9 17:28 RC_2021-05-07  
-rw-r--r-- 1 root root 8725756019 Aug 9 17:48 RC_2021-05-08  
-rw-r--r-- 1 root root 8889488493 Aug 9 18:07 RC_2021-05-09  
-rw-r--r-- 1 root root 9605987029 Aug 9 18:24 RC_2021-05-10  
-rw-r--r-- 1 root root 9938707285 Aug 9 18:43 RC_2021-05-11  
-rw-r--r-- 1 root root 10076510269 Aug 9 19:02 RC_2021-05-12  
-rw-r--r-- 1 root root 9883018150 Aug 9 19:19 RC_2021-05-13  
-rw-r--r-- 1 root root 9695352031 Aug 9 19:36 RC_2021-05-14  
-rw-r--r-- 1 root root 8726999970 Aug 9 19:52 RC_2021-05-15  
-rw-r--r-- 1 root root 9160705762 Aug 9 20:09 RC_2021-05-16  
-rw-r--r-- 1 root root 10034858757 Aug 9 20:31 RC_2021-05-17  
-rw-r--r-- 1 root root 10085444956 Aug 9 20:58 RC_2021-05-18  
-rw-r--r-- 1 root root 10223552907 Aug 9 21:28 RC_2021-05-19  
-rw-r--r-- 1 root root 10035523908 Aug 9 21:49 RC_2021-05-20  
-rw-r--r-- 1 root root 9366915647 Aug 9 22:14 RC_2021-05-21  
-rw-r--r-- 1 root root 8595795622 Aug 10 00:27 RC_2021-05-22  
-rw-r--r-- 1 root root 8821664968 Aug 10 00:44 RC_2021-05-23  
-rw-r--r-- 1 root root 9292102711 Aug 10 01:07 RC_2021-05-24  
drwxr-xr-x 2 root root 4096 Aug 10 01:07 .  
-rw-r--r-- 1 root root 7022061644 Aug 10 01:28 RC_2021-05-25  
root@es3:/data/reddit#
```

Applying the model

- We apply the model to the numpy matrix representation of the image
- `result` is just a numpy version of `results`
 - This contains four types of information

```
results = centernet(image1_np)
result = {key:value.numpy() for key,value in results.items()}
print(result.keys())
```

```
dict_keys(['detection_scores', 'num_detections', 'detection_boxes', 'detection_classes'])
```

Applying the model

- The below functions are defined out of convenience

```
def top_k_objects(result, k=3):
    top_scores = result['detection_scores'][0][0:k]
    top_ids = [labels[str(int(i))] for i in result['detection_classes'][0][0:k]]
    for row in zip(top_scores, top_ids):
        print('Object: ' + row[1] + ', score: ' + str(row[0]))

def prob_person(result):
    id_person = 1
    if len(np.where(result['detection_classes'][0] == 1)[0]):
        top_person_loc = np.where(result['detection_classes'][0] == 1)[0][0]
        people = np.where(result['detection_classes'][0] == 1)[0]
        max_prob = result['detection_scores'][0][top_person_loc]
        implied_prob = 1 - np.prod(1 - result['detection_scores'][0][people])
        print('Maximum probability of an object in the photo being a person: ' + str(max_prob) + \
              '\nProbability of at least 1 person: ' + str(implied_prob))
    else:
        print('No person found')
```

- The first function reports the top k objects detected, based on the model's weights
- The second function reports the highest probability that a person was included in the image as well as an aggregate probability measure

Analyzing the first image

```
Python | top_k_objects(result, 3)
```

```
Object: tie, score: 0.56596684  
Object: person, score: 0.45707893  
Object: tv, score: 0.3345726
```

```
Python | prob_person(result)
```

```
Maximum probability of an object in the photo  
being a person: 0.45679325  
Probability of at least 1 person:  
0.5256033539772034
```



Applying to the second image

```
results = centernet(image2_np)
result = {key:value.numpy() for key,value in results.items() }
```

```
top_k_objects(result, 3)
```

Object: book, score: 0.7087656
Object: tv, score: 0.10406752
Object: book, score: 0.07747121

```
prob_person(result)
```

No person found

```
-rw-r--r-- 1 root root 9916885161 Aug 9 15:13 RC_2021-04-30
-rw-r--r-- 1 root root 9223343241 Aug 9 15:30 RC_2021-05-01
-rw-r--r-- 1 root root 9646977002 Aug 9 15:48 RC_2021-05-02
-rw-r--r-- 1 root root 9790222766 Aug 9 16:06 RC_2021-05-03
-rw-r--r-- 1 root root 9629653589 Aug 9 16:23 RC_2021-05-04
-rw-r--r-- 1 root root 10128104379 Aug 9 16:43 RC_2021-05-05
-rw-r--r-- 1 root root 10030968634 Aug 9 17:06 RC_2021-05-06
-rw-r--r-- 1 root root 9640296547 Aug 9 17:28 RC_2021-05-07
-rw-r--r-- 1 root root 8725756019 Aug 9 17:48 RC_2021-05-08
-rw-r--r-- 1 root root 8889488493 Aug 9 18:07 RC_2021-05-09
-rw-r--r-- 1 root root 9605987029 Aug 9 18:24 RC_2021-05-10
-rw-r--r-- 1 root root 9938707285 Aug 9 18:43 RC_2021-05-11
-rw-r--r-- 1 root root 10076510269 Aug 9 19:02 RC_2021-05-12
-rw-r--r-- 1 root root 9883018150 Aug 9 19:19 RC_2021-05-13
-rw-r--r-- 1 root root 9695352031 Aug 9 19:36 RC_2021-05-14
-rw-r--r-- 1 root root 8726999970 Aug 9 19:52 RC_2021-05-15
-rw-r--r-- 1 root root 9160705762 Aug 9 20:09 RC_2021-05-16
-rw-r--r-- 1 root root 10034858757 Aug 9 20:31 RC_2021-05-17
-rw-r--r-- 1 root root 10085444956 Aug 9 20:58 RC_2021-05-18
-rw-r--r-- 1 root root 10223552907 Aug 9 21:28 RC_2021-05-19
-rw-r--r-- 1 root root 10035523908 Aug 9 21:49 RC_2021-05-20
-rw-r--r-- 1 root root 9366915647 Aug 9 22:14 RC_2021-05-21
-rw-r--r-- 1 root root 8595795622 Aug 10 00:27 RC_2021-05-22
-rw-r--r-- 1 root root 8821664968 Aug 10 00:44 RC_2021-05-23
-rw-r--r-- 1 root root 9292102711 Aug 10 01:07 RC_2021-05-24
drwxr-xr-x 2 root root 4096 Aug 10 01:07 .
-rw-r--r-- 1 root root 7022061644 Aug 10 01:28 RC_2021-05-25
root@es3:/data/reddit#
```

Try this yourself!

- The code is available in the Jupyter Notebook
- Alternatively, you can go to: <https://colab.research.google.com/drive/1-kUtNNVZtKPMBuGRrJwk0-yQQ0ZsOg2F?usp=sharing>



Video data

Working with video

- Video data is challenging – very storage intensive
 - Ex.: Uber's self driving cars would generate >100GB of data *per hour per car*
- Video data is very promising
 - Think of how many task involve vision!
 - Driving
 - Photography
 - Warehouse auditing...
- At the end of the day though, video is just a sequence of images

One method for video

YOLOv3

- You
- Only
-
- Once

You Only Look Once: Because the algorithm only does one pass (looks once) to classify any and all objects

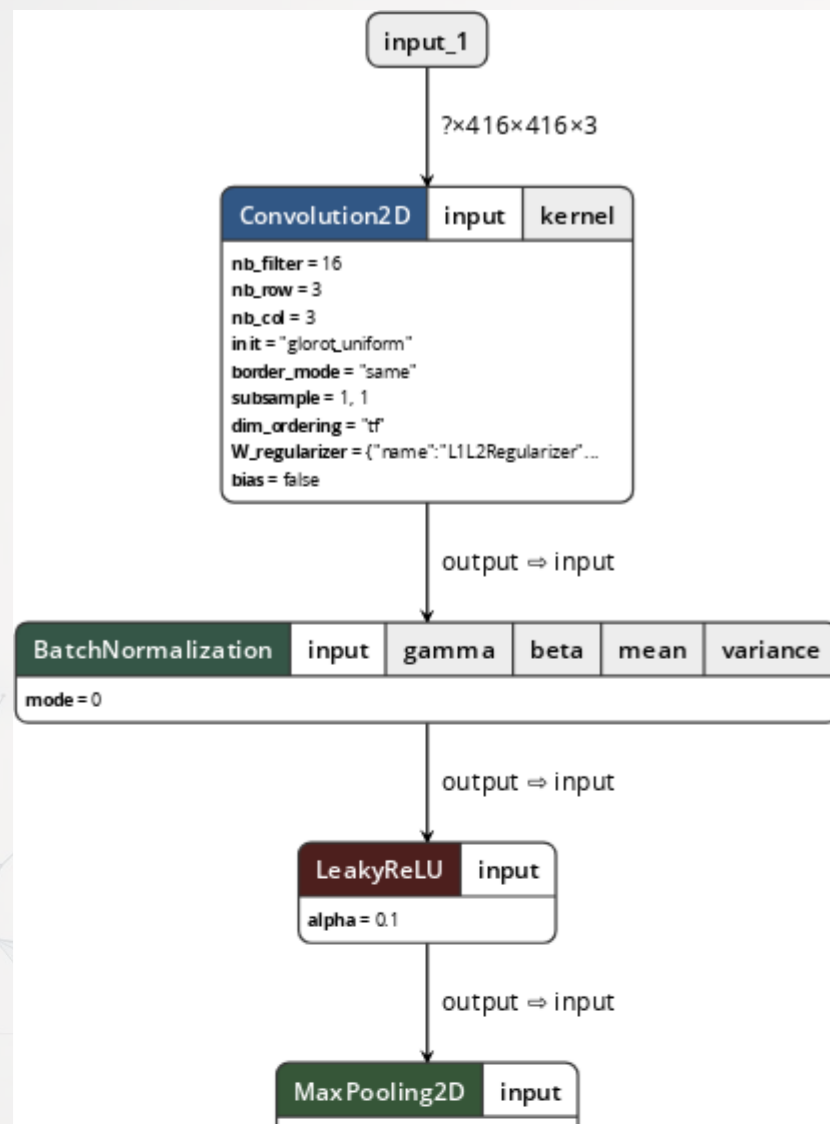


Video link

What does YOLO do?

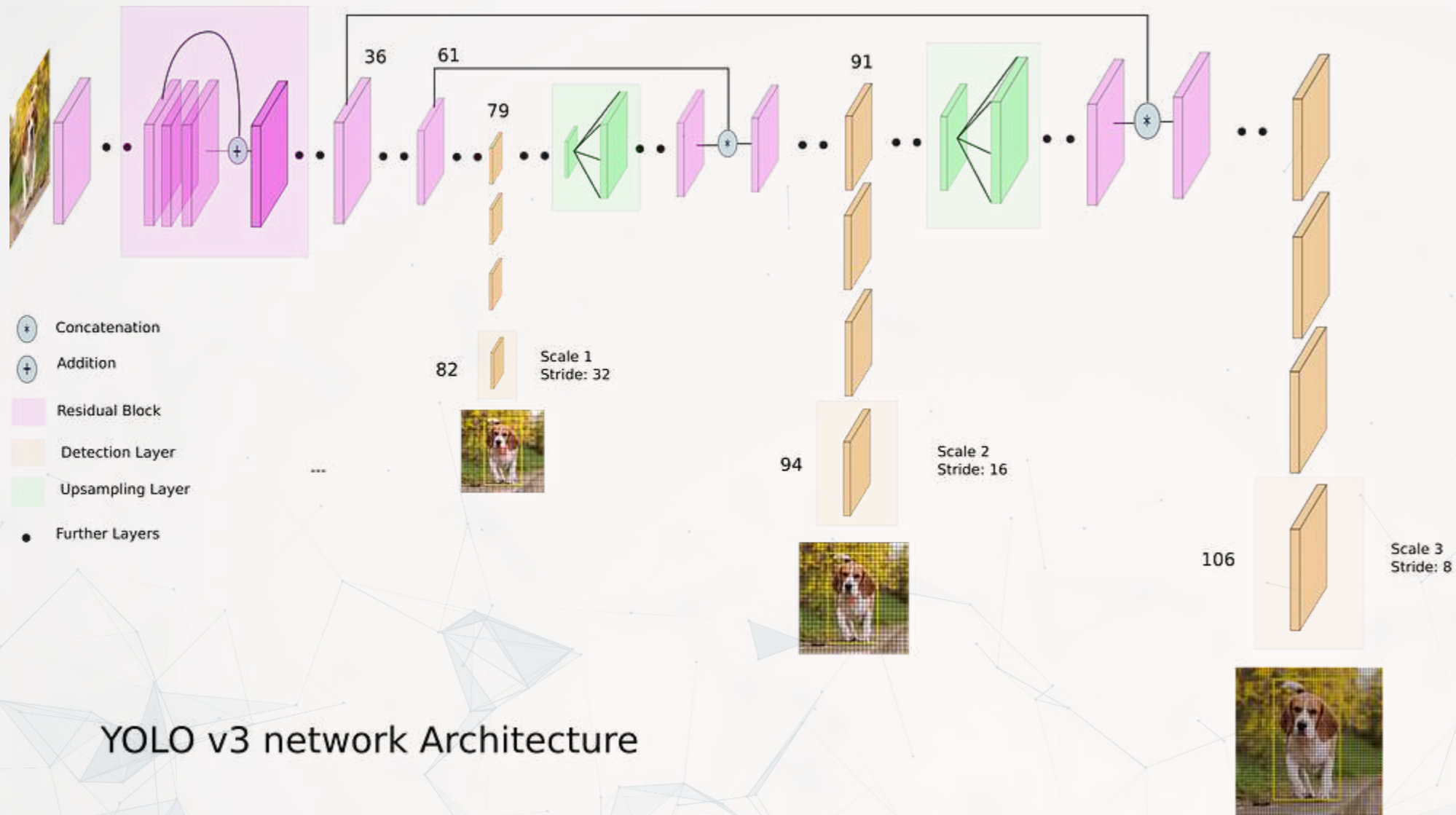
- It spots objects in videos and labels them
 - It also figures out a *bounding box* – a box containing the object inside the video frame
- It can spot *overlapping* objects
- It can spot multiple of the same or different object types
- The baseline model (using the COCO dataset) can detect 80 different object types
 - There are other datasets with more objects

How does Yolo do it? Map of Tiny YOLO



Yolo model and graphing tool from [lutzroeder/netron](https://github.com/lutzroeder/netron)

How does Yolo do it?



YOLO v3 network Architecture

Diagram from *What's new in YOLO v3* by Ayoosh Kathuria

Where to get video data

- One extensive source is [Youtube-8M](#)
 - 6.1M videos, 3-10 minutes each
 - Each video has >1,000 views
 - 350,000 hours of video
 - 237,000 labeled 5 second segments
 - 1.3B video features that are machine labeled
 - 1.3B audio features that are machine labeled

Object detection in practice

- An algorithm like YOLO v3 is somewhat tricky to run
- Preparing the algorithm takes a long time
 - The final output, though, can run on much cheaper hardware
- These algorithms just recently became feasible so their impact has yet to be felt so strongly

Think about how facial recognition showed up everywhere for images over the past few years

A word on ethics of object detection

But maybe a better question is: “What are we going to do with these detectors now that we have them?” A lot of the people doing this research are at Google and Facebook. I guess at least we know the technology is in good hands and definitely won’t be used to harvest your personal information and sell it to.... wait, you’re saying that’s exactly what it will be used for?? Oh.

Well the other people heavily funding vision research are the military and they’ve never done anything horrible like killing lots of people with new technology oh wait....¹

I have a lot of hope that most of the people using computer vision are just doing happy, good stuff with it, like counting the number of zebras in a national park [13], or tracking their cat as it wanders around their house [19]. But computer vision is already being put to questionable use and as researchers we have a responsibility to at least consider the harm our work might be doing and think of ways to mitigate it. We owe the world that much.

In closing, do not @ me. (Because I finally quit Twitter).

¹The author is funded by the Office of Naval Research and Google.

From Redmon and Farhadi (2018) [The YOLO v3 paper]



Combining images and text

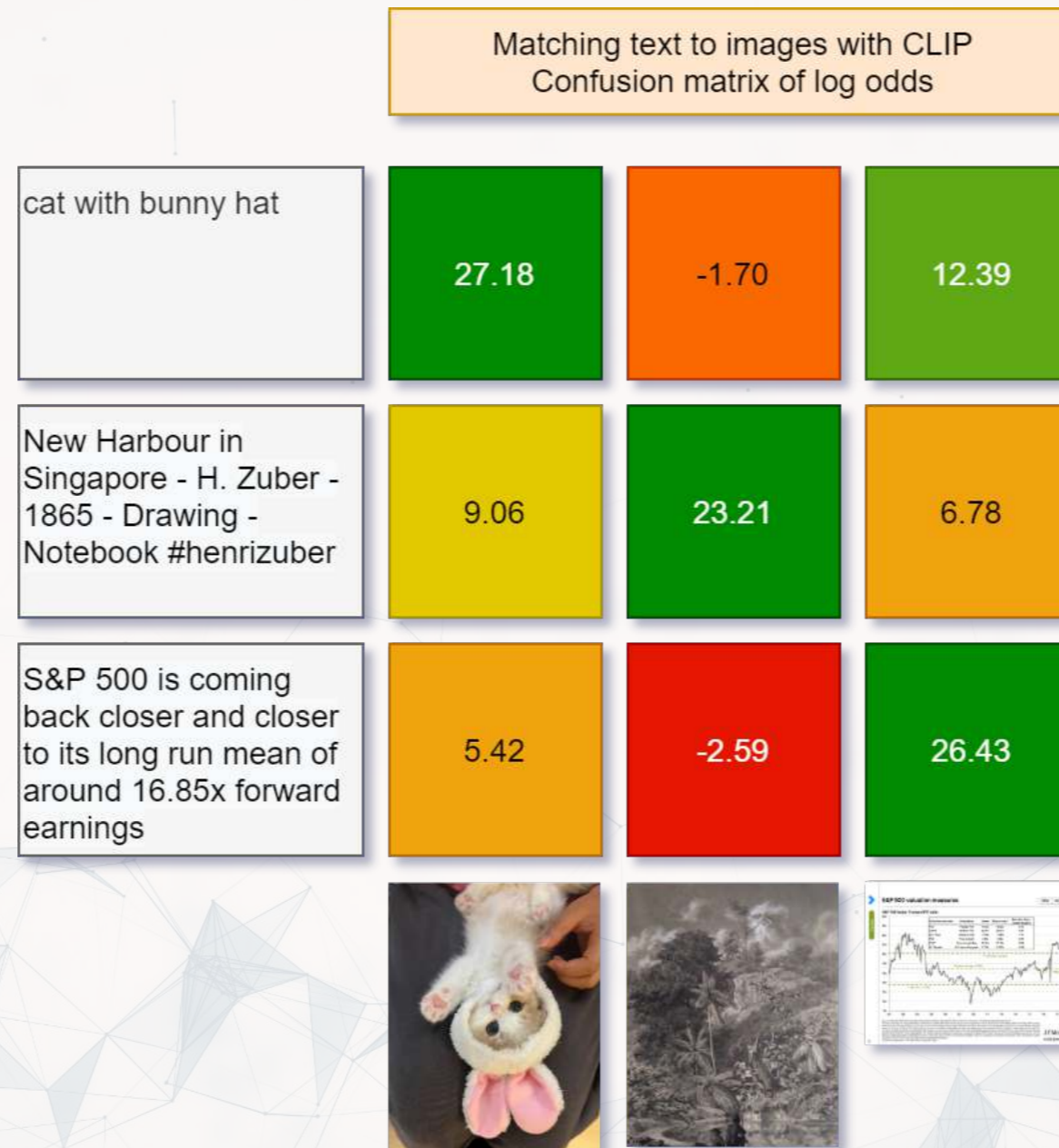
Large language models + Images

- Multiple impactful models were released since 2021 that merge text and image processing into a single model
 - CLIP: Contrastive Language-Image Pre-training
 - Pairs images with captions
 - Stable Diffusion
 - Image generation from text

These work by embedding images and text into the same embedding space

CLIP

- Code for this is available at: rmlink.com/colab_clip



Stable diffusion: Content

- Code to implement as a Telegram bot:
<https://github.com/rmcrowley2000/StableDiffBot>

“A photo of the Singapore skyline including Marina Bay Sands”

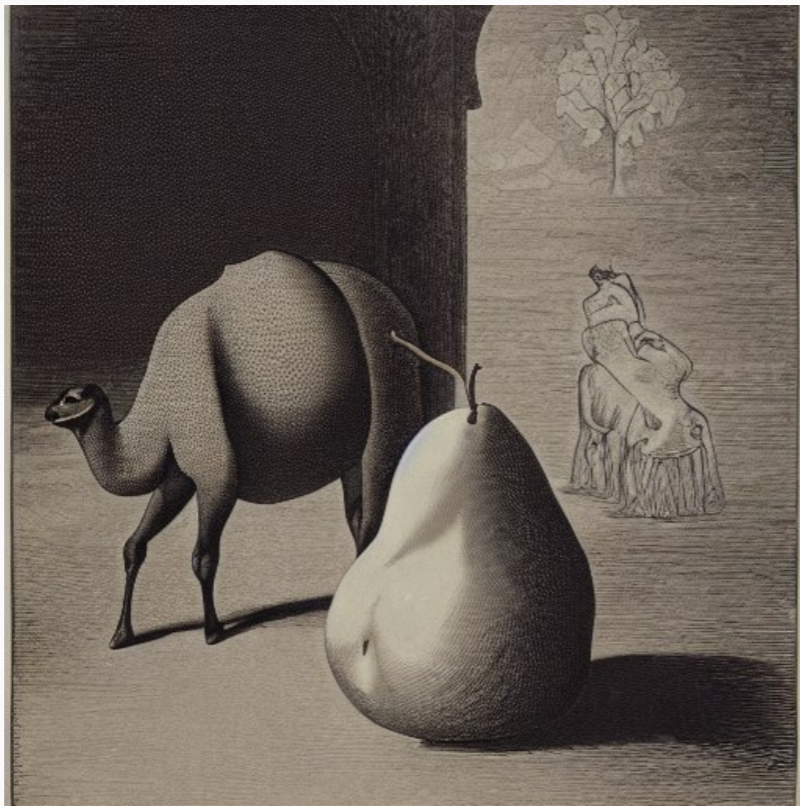


“Singapore Management University”

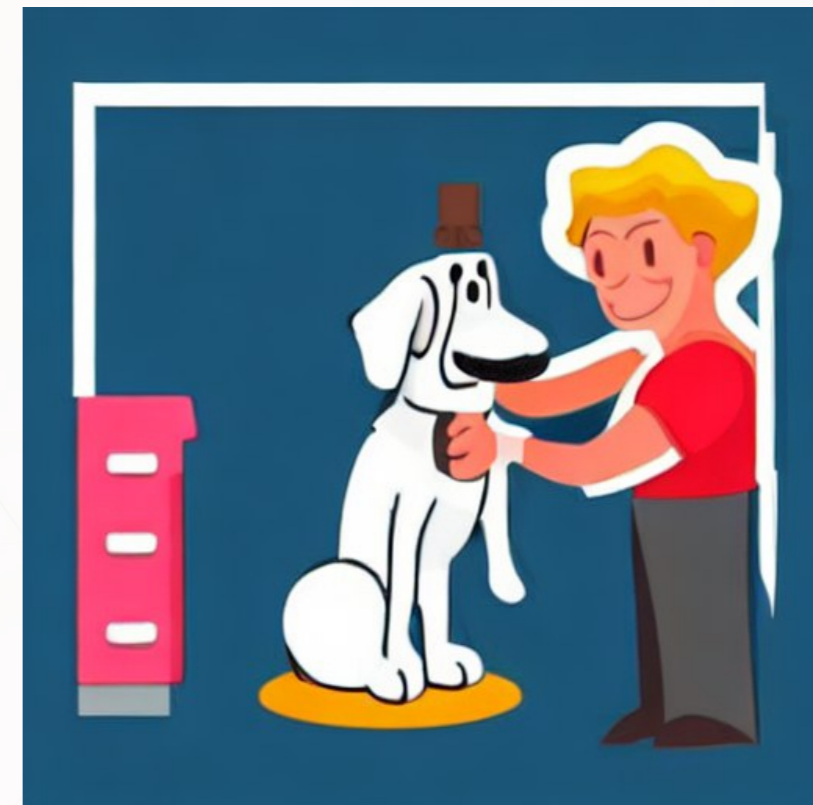


Stable diffusion: Style

“Lithograph of a camel eating a pear”

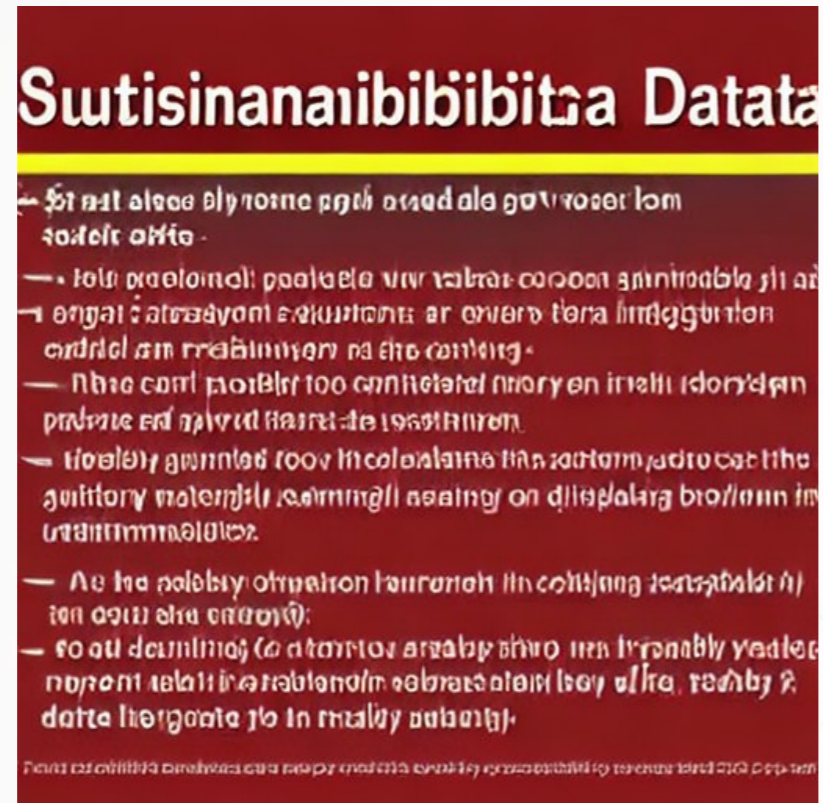


“A cartoon icon of a dog getting a hair cut.”



Stable diffusion: Problems

“Sustainability data”

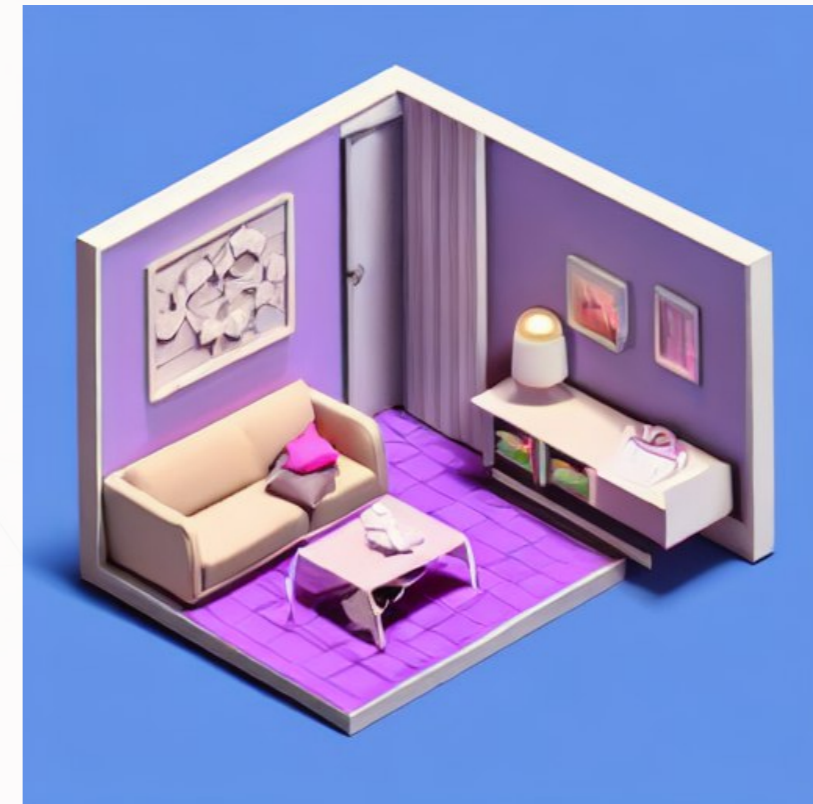
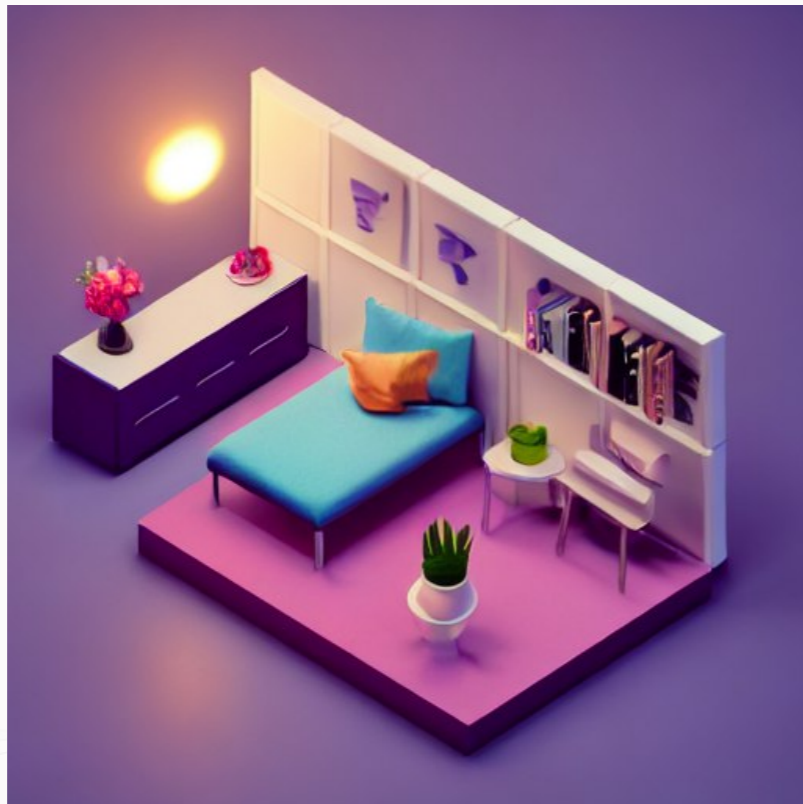


“A cavapoo enjoying a nice warm cup of tea”



Stable diffusion: Complexity

“Tiny cute isometric living room in a cutaway box, soft smooth lighting, soft colors, purple and blue color scheme, soft colors, 100mm lens, 3d blender render”





Conclusion

Wrap-up

Neural networks can accurately classify entire images

- Useful for clustering our classifying images

Neural networks can accurately classify or detect objects included in images

- Opens up a lot of possibilities
 - Such as looking at whether a person is wearing a mask or not (related to HW3)

Neural networks can combine text and images into a measure

- Opens up more empirical possibilities and new ways to use image data

What remains

- Assignment 2
 - New due date: **TBD**
 - You are welcome to submit earlier
- Assignment 3
 - Shorter than the other assignments
 - Focuses on image detection and classification
 - It's all done on Colab so that you don't need to worry about getting pytorch to work locally
 - Due: **December TBD**
- Proposal
 - Due: **December TBD**

Can't extend past December TBD due to grade deadlines

Going forward

If you want to discuss any research ideas down the line involving ML or alternative data (text, image, video), you are welcome to reach out to me

- I also maintain a large collection of Twitter data
 - If you need data down the line, you can reach out to me
 - Company tweets (US, 15+ other countries)
 - CEO and CFO tweets (US)
 - COVID-19 tweets (global, multilingual, geotagged)
 - WallStreetBets tweets
 - Ukraine-Russia war tweets
 - All data includes tweet characteristics, user characteristics, images, links, text, and derived features (content, sentiment, emotion, grammar)

Packages used for these slides

Python

- matplotlib
- numpy
- pandas
- PIL
- requests
- seaborn
- shap
- tensorflow
- tensorflow_gan
- tensorflow_hub
- transformers

R

- kableExtra
- knitr
- quarto
- reticulate
- revealjs

References

- Aubry, Mathieu, Roman Kraeusel, Gustavo Manso, and Christophe Spaenjers. “Biased auctioneers.” *Journal of Finance*, Forthcoming (2022).
- Liu, Liu, Daria Dzyabura, and Natalie Mizik. “Visual listening in: Extracting brand image portrayed on social media.” *Marketing Science* 39, no. 4 (2020): 669-686.
- Redmon, Joseph, and Ali Farhadi. “YOLOv3: An Incremental Improvement.” arXiv, April 8, 2018. <http://arxiv.org/abs/1804.02767>.
- Yasrab, Robail, Naijie Gu, and Xiaoci Zhang. “An encoder-decoder based convolution neural network (CNN) for future advanced driver assistance system (ADAS).” *Applied Sciences* 7, no. 4 (2017): 312.
- Zhang, Shunyuan, Dokyun DK Lee, Param Vir Singh, and Kannan Srinivasan. “How much is an image worth? Airbnb property demand estimation leveraging large scale image analytics.” *Airbnb Property Demand Estimation Leveraging Large Scale Image Analytics* (May 25, 2021) (2021).
- Zhou, Xingyi, Dequan Wang, and Philipp Krähenbühl. “Objects as points.” arXiv preprint arXiv:1904.07850 (2019).

