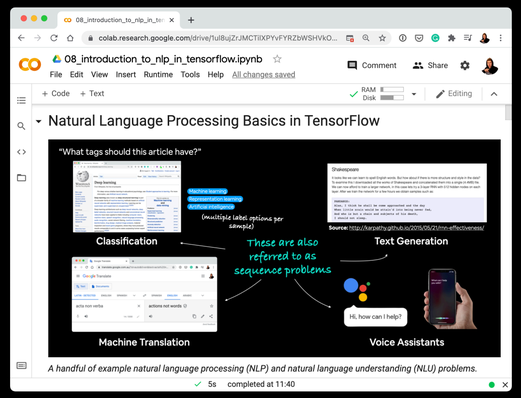
Natural Language Processing (NLP) with

TensorFlow

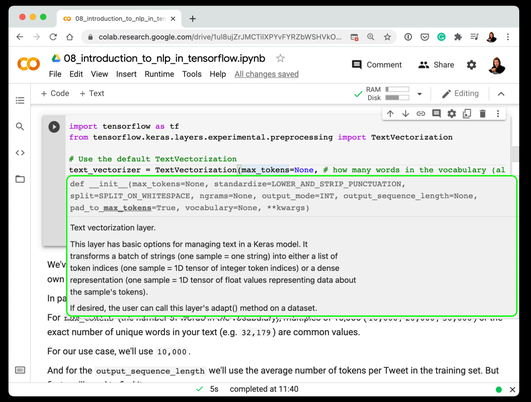
Where can you get help?

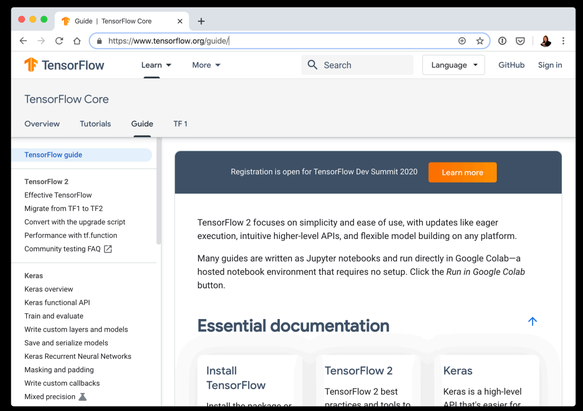
- Follow along with the code
- Try it for yourself
- Press SHIFT + CMD + SPACE to read the docstring
- Search for it
- Try again
- Ask (don't forget the Discord chat!)

(yes, including the "dumb" questions)



"If in doubt, run the code"



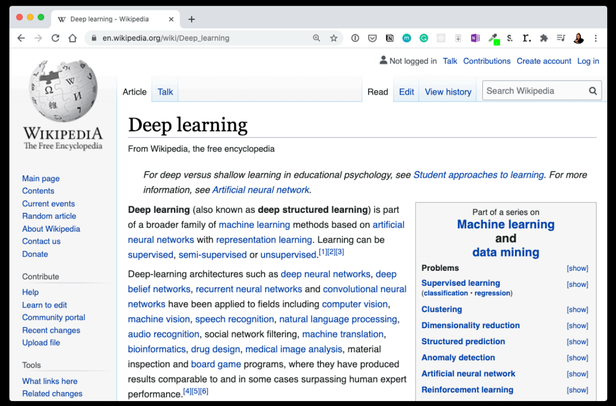


“What is a NLP problem?”

Example NLP problems

and NLU... (natural language understanding)

"What tags should this article have?"



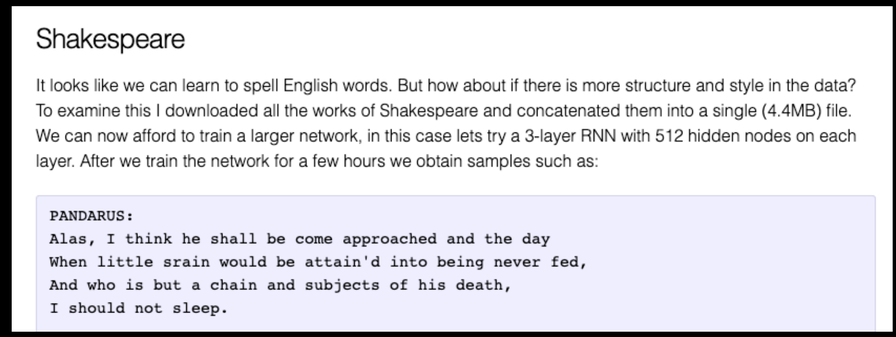
Machine learning

Representation learning

Artificial intelligence

(multiple label options per sample)

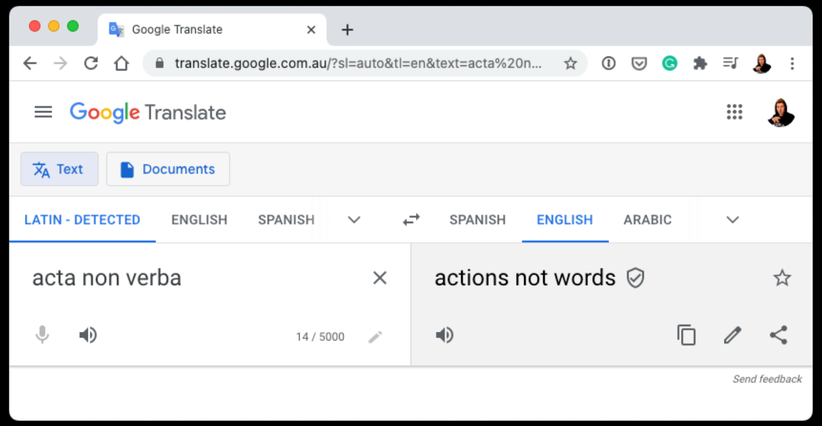
Classification



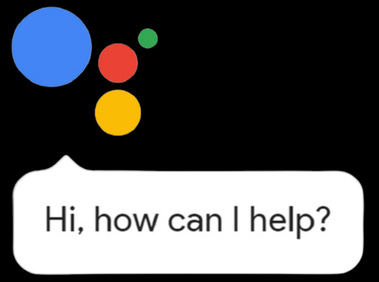
Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Text Generation

These are also referred to as sequence problems

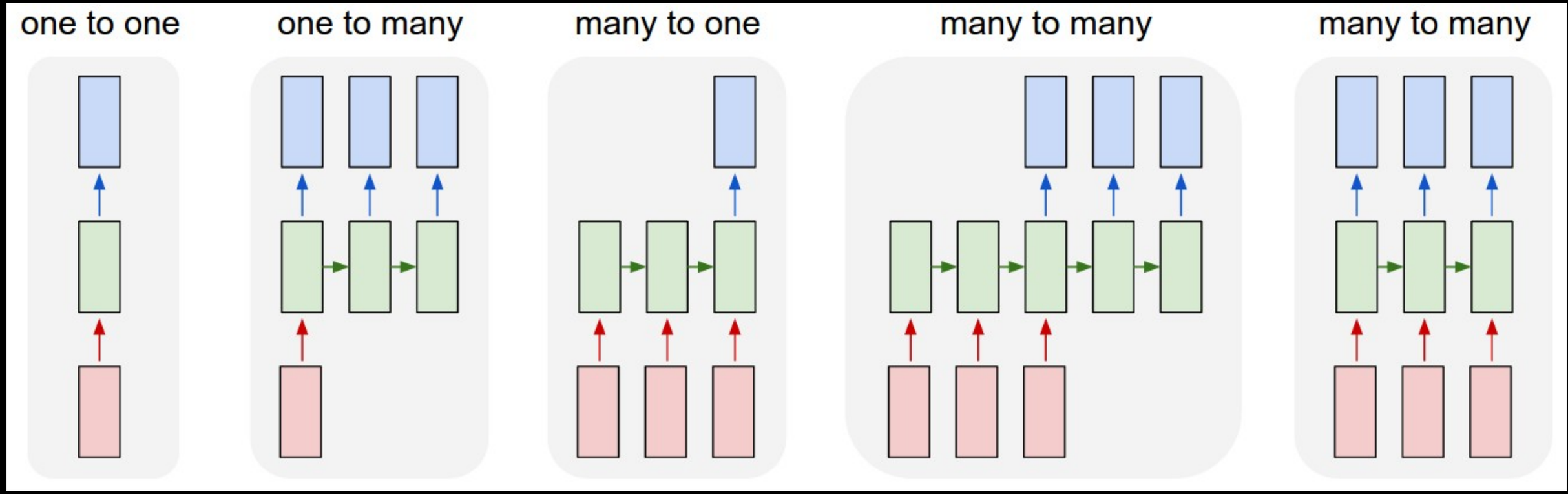


Machine Translation



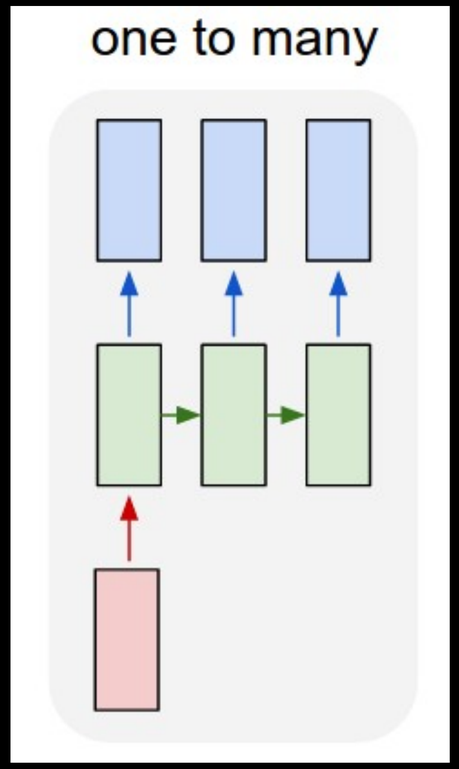
Voice Assistants

Other sequence problems



Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Other sequence problems



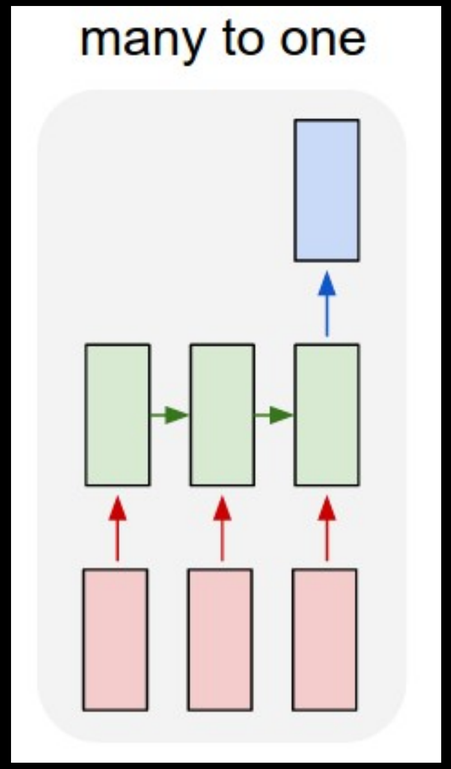
Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Image captioning

Input

Output A sledgehammer leaning up against a tire

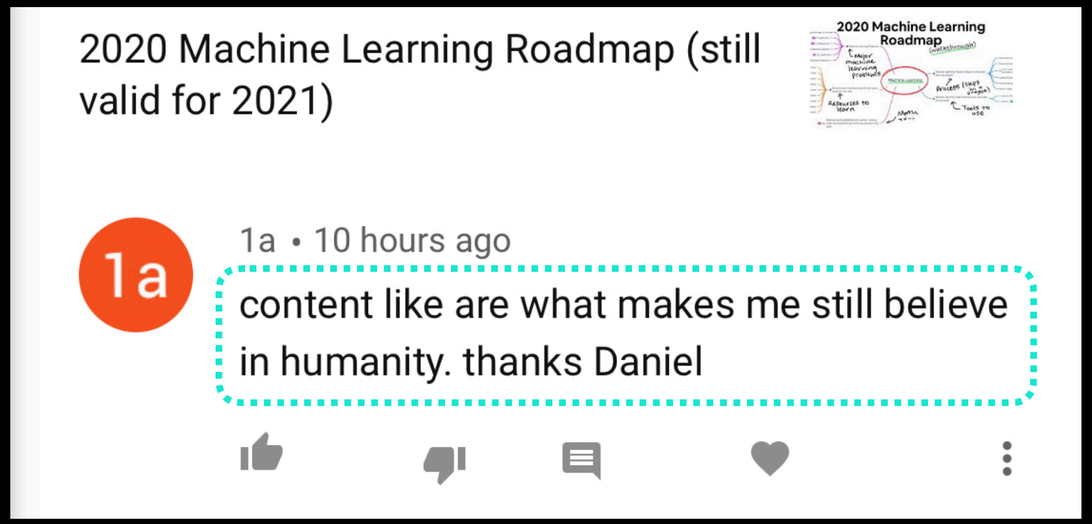
Other sequence problems



Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Input

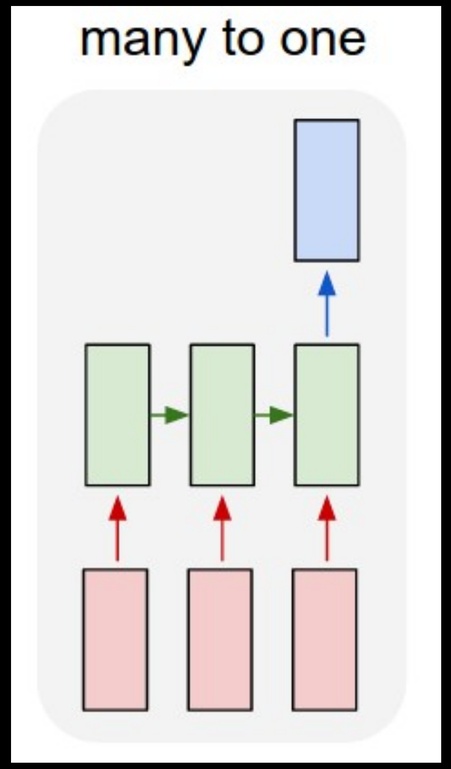
Sentiment analysis



Output

Positive 👍

Other sequence problems



Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Input

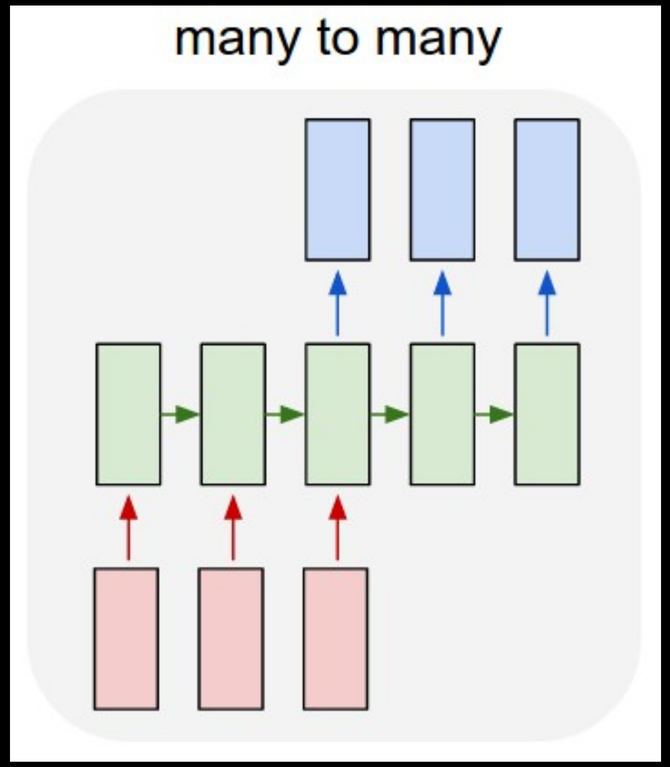
Time series forecasting

Source: https://www.coindesk.com/price/bitcoin

Output

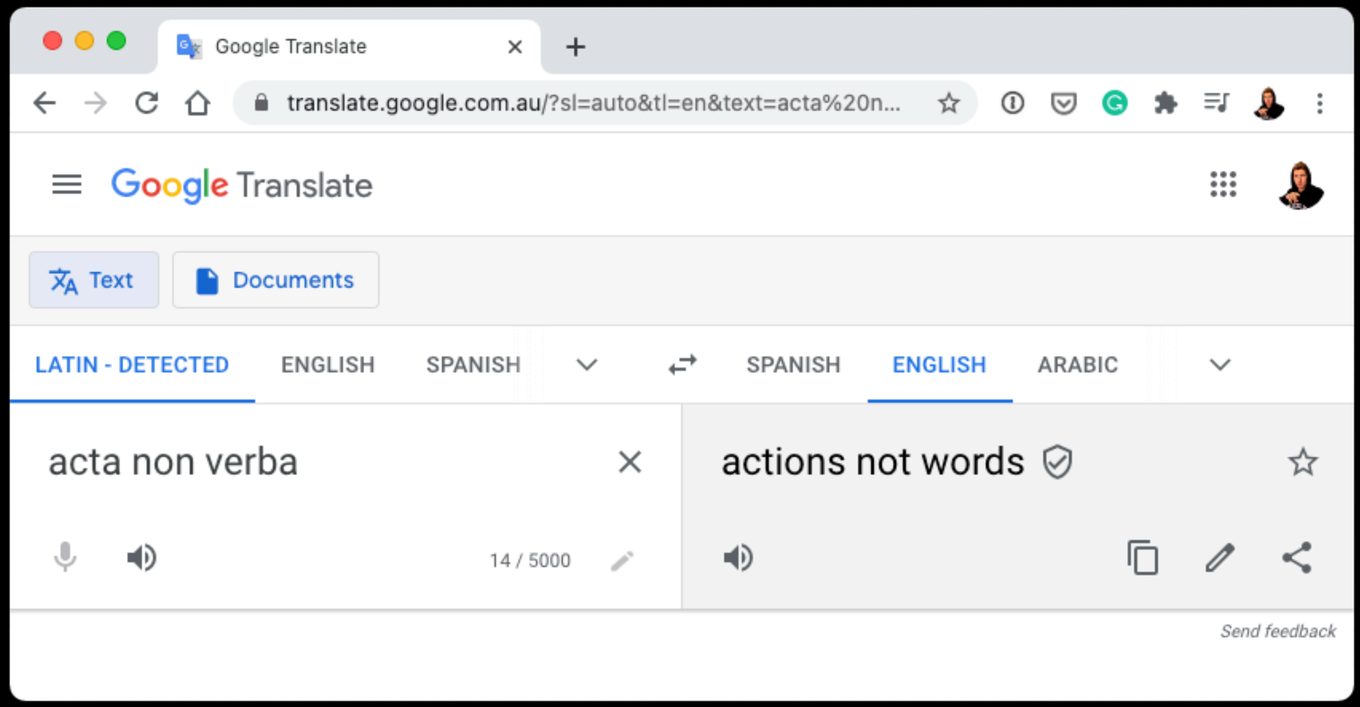
Price at next timestamp (e.g. $59,678)

Other sequence problems



Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Machine Translation



Input

Output

What we're going to cover

(broadly)

- Downloading and preparing a text dataset
- How to prepare text data for modelling (tokenization and embedding)
- Setting up multiple modelling experiments with recurrent neural networks (RNNs)
- Building a text feature extraction model using TensorFlow Hub
- Finding the most wrong prediction examples
- Using a model we've built to make predictions on text from the wild

(we'll be cooking up lots of code!)

How:

NLP inputs and outputs

"Is this Tweet for a disaster or not?"

Diaster 🌪️

Not Diaster 👌

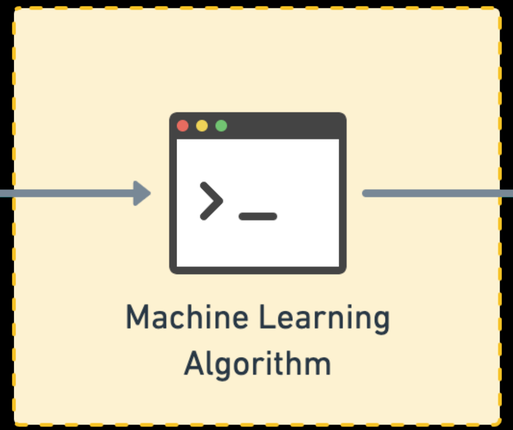
Actual output

```
[[0.22, 0.98, 0.02…],
[0.09, 0.55, 0.87…],
[0.53, 0.81, 0.79…],
…,
```

Numerical encoding

(Tokenization + Embedding)

Inputs



(often already exists, if not, you can build one)

Outputs

🌪️ 👌

```
[[0.97, 0.03],
[0.81, 0.19],
…,
```

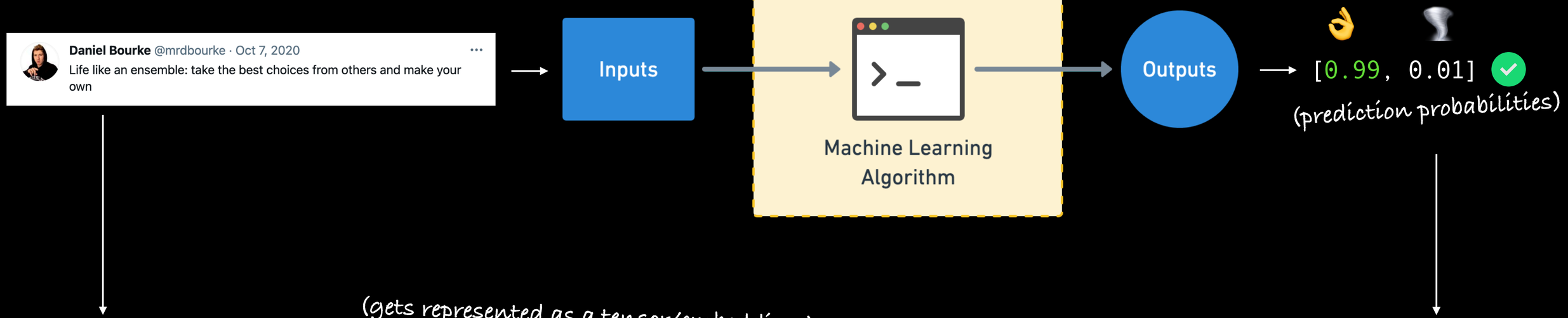
Predicted output

(comes from looking at lots of these)

Input and output shapes

(for a text classification example)

We're going to be building RNNs/ CNNs/Feature extractors to do this part!



(gets represented as a tensor/embedding)

`[batch_size, embedding_size]`

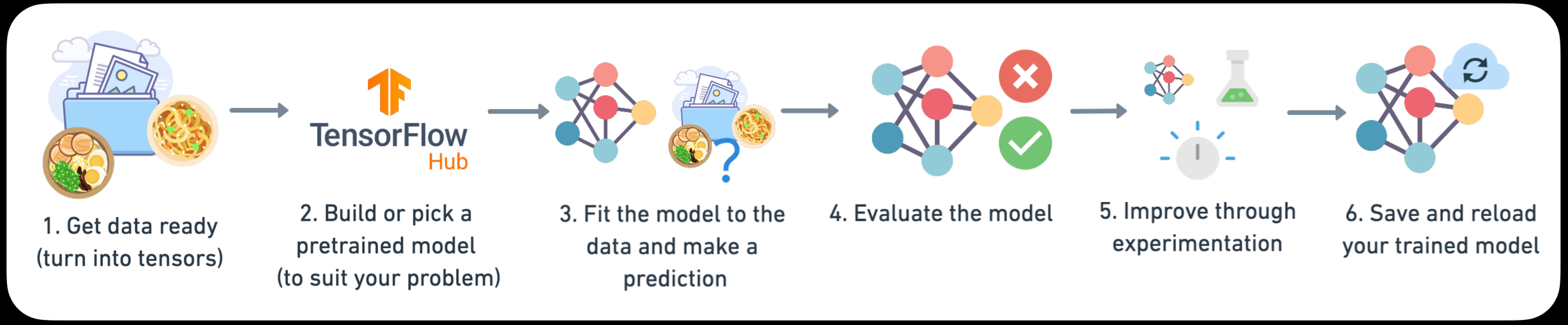
```
Shape = [None, 512]
        or
Shape = [32, 512]
```

(32 is a very common batch size)

These will vary depending on the problem you're working on/what embedding style you use.

```
Shape = [2]
```

Steps in modelling with TensorFlow



1. Turn all data into numbers (neural networks can't handle text/natural language)
2. Make sure all of your tensors are the right shape (pad sequences which don't fit)

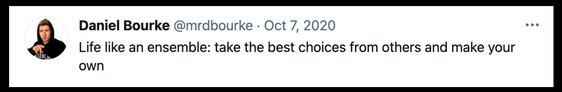
“What is a recurrent neural network (RNN)?”

(typical)*

Architecture of an RNN

Hyperparameter/Layer type	What does it do?	Typical values
Input text(s)	Target texts/sequences you'd like to discover patterns in	Whatever you can represent as text or a sequence
Input layer	Takes in target sequence	`input_shape = [batch_size, embedding_size]` or `[batch_size, sequence_shape]`
Text vectorization layer	Maps input sequences to numbers	Multiple, can create with `tf.keras.layers.experimental.preprocessing.TextVectorization`
Embedding	Turns mapping of text vectors to embedding matrix (representaiton of how words relate)	Multiple, can create with `tf.keras.layers.Embedding`
RNN cell(s)	Finds patterns in sequences	SimpleRNN, LSTM, GRU
Hidden activation	Adds non-linearity to learned features (non-straight lines)	Usually Tanh (hyperbolic tangent) (`tf.keras.activations.tanh`)
Pooling layer	Reduces the dimensionality of learned sequence features (usually for Conv1D models)	Average (`tf.keras.layers.GlobaelAveragePooling1D`) or Max (`tf.keras.layers.GlobalMaxPool1D`)
Fully connected layer	Further refines learned features from recurrent layers	`tf.keras.layers.Dense`
Output layer	Takes learned features and outputs them in shape of target labels	`output_shape = [number_of_classes]` (e.g. 2 for Diaster, Not Diaster)
Output activation	Adds non-linearities to output layer	`tf.keras.activations.sigmoid` (binary classification) or `tf.keras.activations.softmax`

(what we're working towards building)



→

```
# 1. Create LSTM model
from tensorflow.keras import layers
inputs = layers.Input(shape=(1,), dtype="string")
x = text_vectorizer(inputs) # turn input sequence to numbers
x = embedding(x) # create embedding matrix
x = layers.LSTM(64, activation="tanh")(x) # return vector for whole sequence
outputs = layers.Dense(1, activation="sigmoid")(x) # create output layer
model = tf.keras.Model(inputs, outputs, name="LSTM_model")

# 2. Compile model
model.compile(loss=tf.keras.losses.BinaryCrossentropy(),
              optimizer=tf.keras.optimizers.Adam(),
              metrics=["accuracy"])

# 3. Fit the model
history = model.fit(train_sentences, train_labels, epochs=5)
```

→ Not Diaster 👌

***Note:** there are almost an unlimited amount of ways you could stack together a recurrent neural network, this slide demonstrates only one.

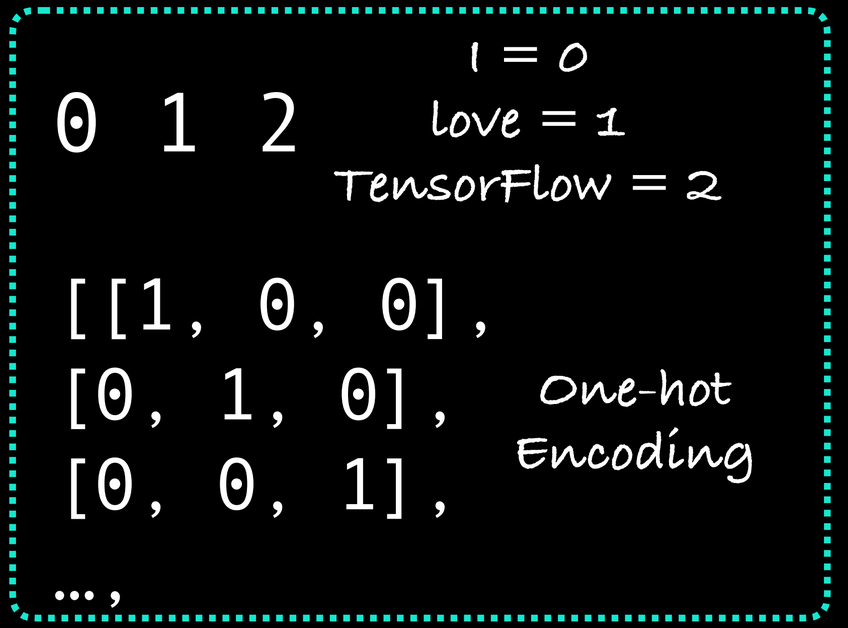
Let’s code!

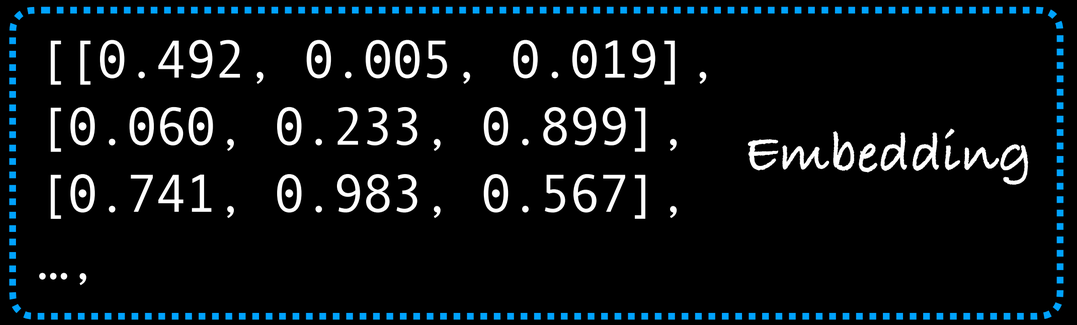
Tokenization vs Embedding

I love TensorFlow

Tokenization — straight mapping from token to number (can be modelled but quickly gets too big)

Embedding — richer representation of relationships between tokens (can limit size + can be learned)





Experiments we're running

Experiment Number	Model
0	Naive Bayes with TF-IDF encoder (baseline)
1	Feed-forward neural network (dense model)
2	LSTM (RNN)
3	GRU (RNN)
4	Bidirectional-LSTM (RNN)
5	1D Convolutional Neural Network
6	TensorFlow Hub Pretrained Feature Extractor
7	TensorFlow Hub Pretrained Feature Extractor (10% of data)

(some common)

Classification evaluation methods

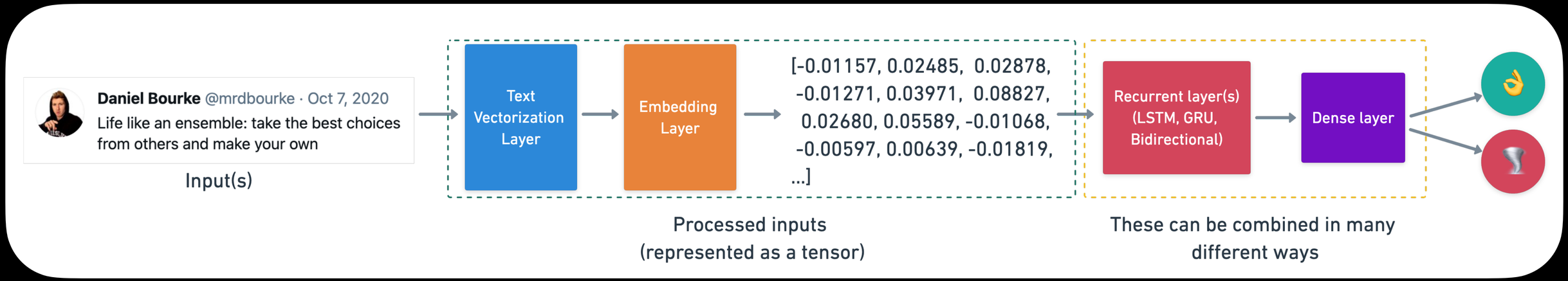
Key: **tp** = True Positive, **tn** = True Negative, **fp** = False Positive, **fn** = False Negative

Metric Name	Metric Forumla	Code	When to use
Accuracy	$\textbf{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$	`tf.keras.metrics.Accuracy()` or `sklearn.metrics.accuracy_score()`	Default metric for classification problems. Not the best for imbalanced classes.
Precision	$\textbf{Precision} = \frac{tp}{tp + fp}$	`tf.keras.metrics.Precision()` or `sklearn.metrics.precision_score()`	Higher precision leads to less false positives.
Recall	$\textbf{Recall} = \frac{tp}{tp + fn}$	`tf.keras.metrics.Recall()` or `sklearn.metrics.recall_score()`	Higher recall leads to less false negatives.
F1-score	$\textbf{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$	`sklearn.metrics.f1_score()`	Combination of precision and recall, usually a good overall metric for a classification model.
Confusion matrix	NA	`Custom function` or `sklearn.metrics.confusion_matrix()`	When comparing predictions to truth labels to see where model gets confused. Can be hard to use with large numbers of classes.

Architecture of a RNN

(coloured block edition)

Standard RNN

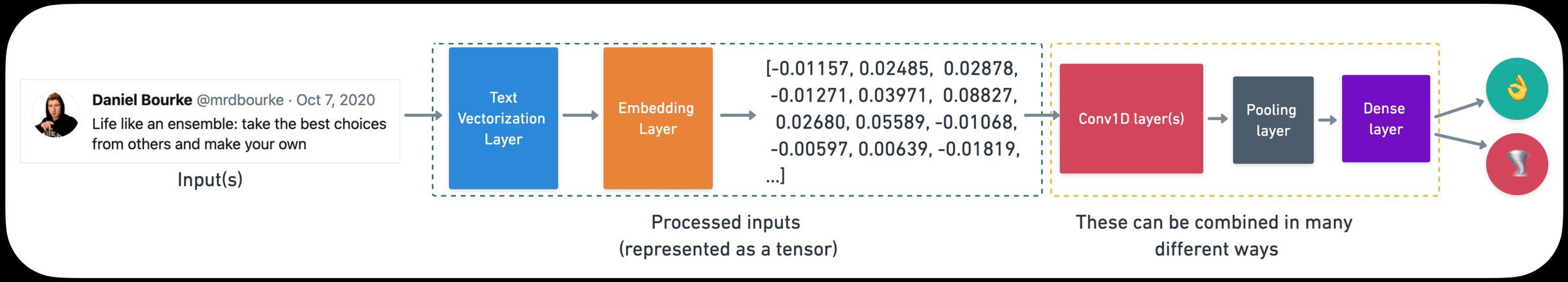


Types of RNN cells

Name	When to use	Learn more	Code
LSTM (long short-term memory)	Default RNN layer for sequence problems.	Understanding LSTM Networks by Chris Olah	`tf.keras.layers.LSTM`
GRU (gated recurrent unit)	Performs very similar to LSTM (could be used as a default).	Illustrated Guide to LSTM's and GRU's by Michael Phi	`tf.keras.layers.GRU`
Bidirectional LSTM (goes forward and backwards on sequence)	Good for sequences which may benefit from passing forward and backwards (e.g. translation or longer passages of text).	Same as above	`tf.keras.layers.Bidirectional`

Architecture of a Sequence Conv1D Model *(coloured block edition)*

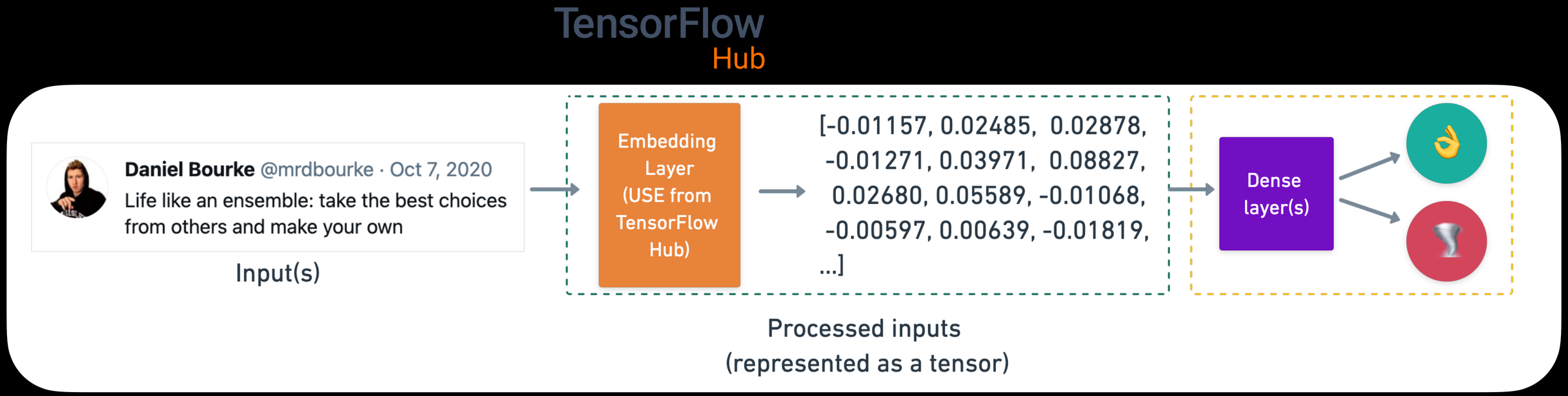
Conv1D Sequence model



Model we're building (USE* feature extractor)

*USE = Universal Sentence Encoder

Source: https://tfhub.dev/google/universal-sentence-encoder/4



Encoder

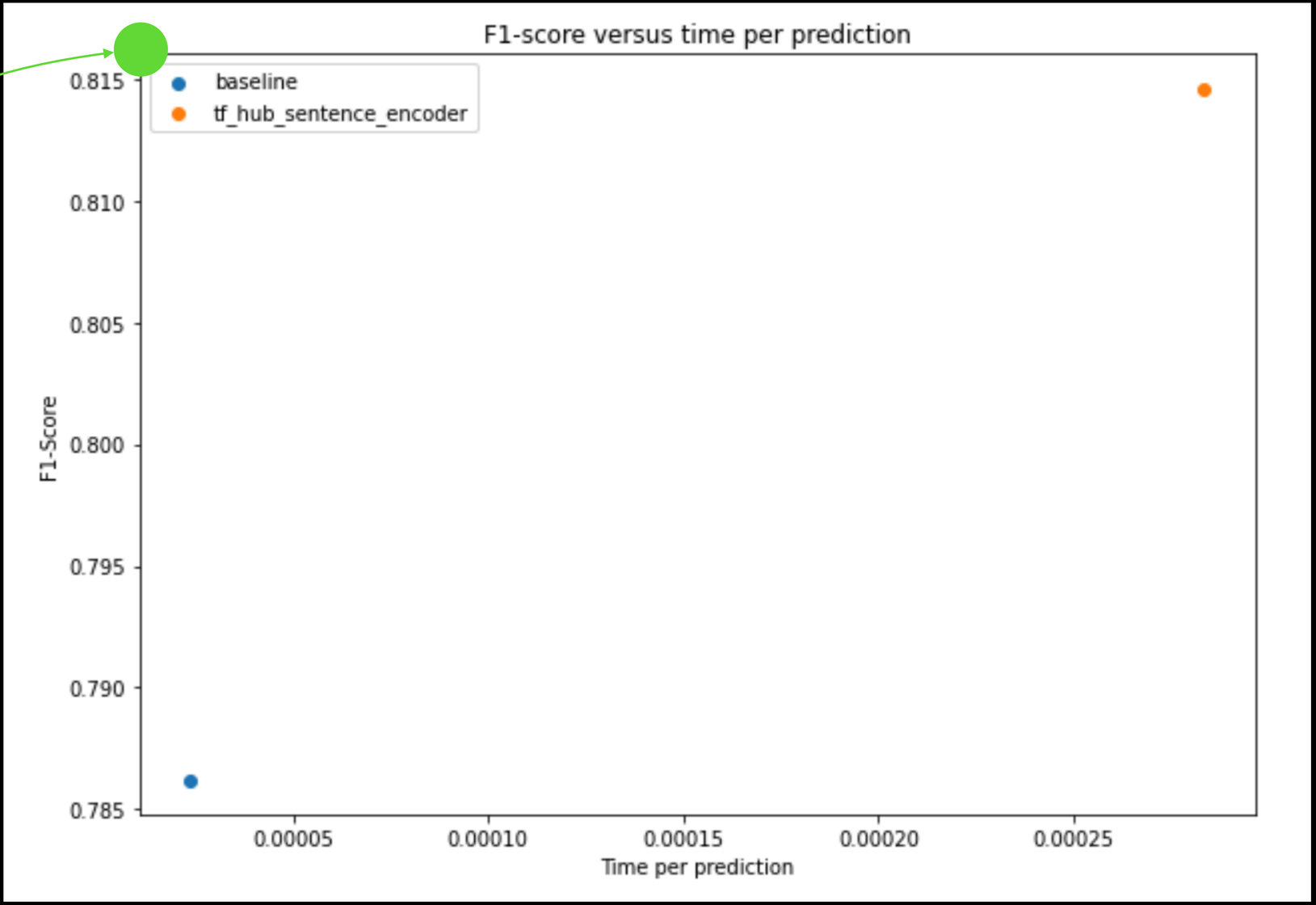
(Encodes sequences into numerical representation)

Decoder

(Decodes sequences into desired output)

Ideal speed/performance trade off

Ideal position for speed/performance (high performance + high speed)



Improving a model

(from a model's perspective)

```python
# 1. Create the model (specified to your problem)
model = tf.keras.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(4, activation="relu"),
  tf.keras.layers.Dense(10, activation="softmax")
])

# 2. Compile the model
model.compile(loss=tf.keras.losses.BinaryCrossentropy(),
              optimizer=tf.keras.optimizers.Adam(lr=0.001),
              metrics=["accuracy"])

# 3. Fit the model
model.fit(X_train_subset, y_train_subset, epochs=5)
```

Smaller model


```python
# 1. Create the model (specified to your problem)
model = tf.keras.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(10, activation="softmax")
])

# 2. Compile the model
model.compile(loss=tf.keras.losses.BinaryCrossentropy(),
              optimizer=tf.keras.optimizers.Adam(lr=0.0001),
              metrics=["accuracy"])

# 3. Fit the model
model.fit(X_train_full, y_train_full, epochs=100)
```

Larger model

Common ways to improve a deep model:

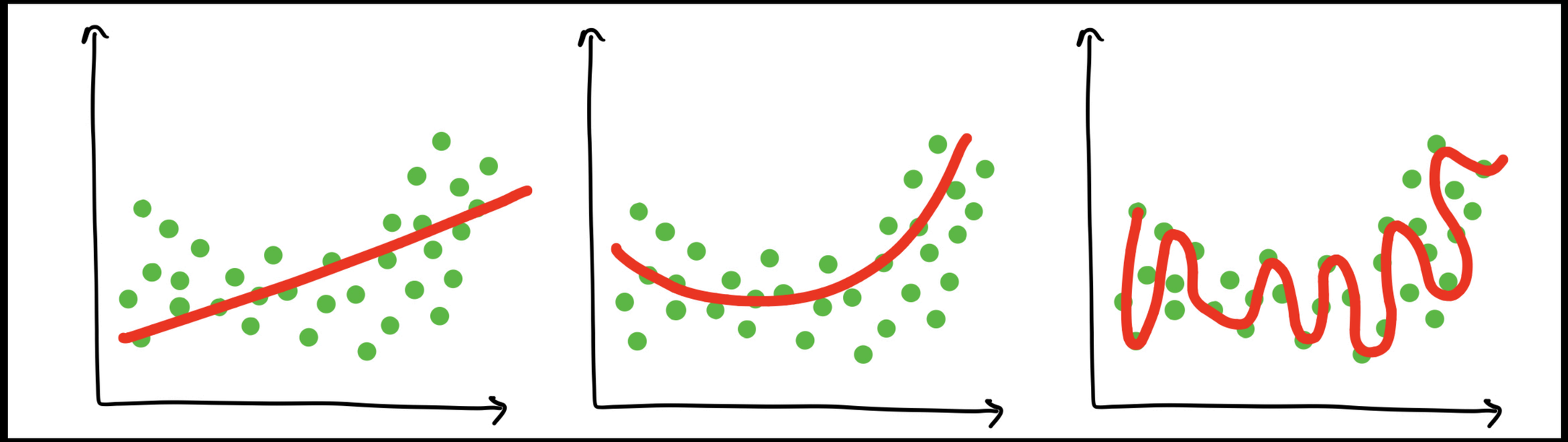
- Adding layers
- Increase the number of hidden units
- Change the activation functions
- Change the optimization function
- Change the learning rate
- Fitting on more data
- Fitting for longer

(because you can alter each of these, they're hyperparameters)

What is overfitting?

Overfitting — when a model over learns patterns in a particular dataset and isn't able to generalise to unseen data.

For example, a student who studies the course materials too hard and then isn't able to perform well on the final exam. Or tries to put their knowledge into practice at the workplace and finds what they learned has nothing to do with the real world.



Underfitting | **Balanced** (goldilocks zone) | **Overfitting**

Improving a model

(from a data perspective)

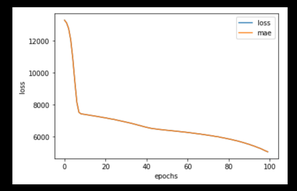
Method to improve a model (reduce overfitting)	What does it do?
More data	Gives a model more of a chance to learn patterns between samples (e.g. if a model is performing poorly on images of pizza, show it more images of pizza).
Data augmentation (usually for images)	Increase the diversity of your training dataset without collecting more data (e.g. take your photos of pizza and randomly rotate them 30°). Increased diversity forces a model to learn more generalisation patterns.
Better data	Not all data samples are created equally. Removing poor samples from or adding better samples to your dataset can improve your model's performance.
Use transfer learning	Take a model's pre-learned patterns from one problem and tweak them to suit your own problem. For example, take a model trained on pictures of cars to recognise pictures of trucks.

The machine learning explorer's motto

"Visualize, visualize, visualize"

Data

Model



Training

Predictions

It's a good idea to visualize these as often as possible.

The machine learning practitioner's motto

"Experiment, experiment, experiment"


(try lots of things and see what tastes good)