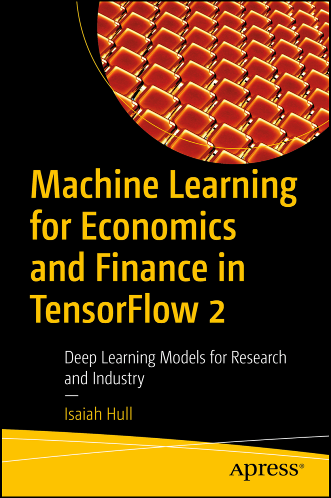
Machine Learning for Economics and Finance in TensorFlow 2

Isaiah Hull

Sveriges Riksbank

June 9, 2021

Introduction



Introduction to TensorFlow

Tutorial Overview

1. TensorFlow
2. Structured Data
3. Unstructured Data
4. GANs
5. Live Coding
6. Q&A

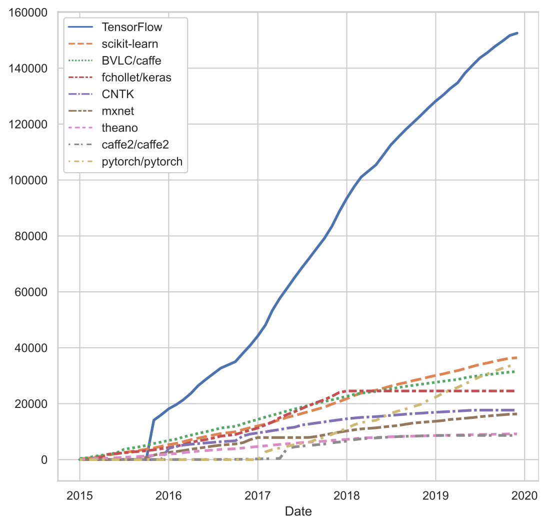
Introduction to TensorFlow

Introduction to TensorFlow

TensorFlow Overview

- **Open source framework for ML introduced by Google's Brain Team.**
  - Large community, tools for production settings, capacity for distributed training.
- **Built for neural networks, but can be used for any graph-based model.**
  - Tree-based models, theoretical models in economics and finance, reinforcement learning.

Introduction to TensorFlow



GitHub stars by ML framework (Perrault et al., 2019).

Introduction to TensorFlow

<u>TensorFlow for Economics and Finance</u>

1. Causal Inference
2. Feature Extraction
3. Non-linear Modeling
4. Simulation
5. Dimensionality Reduction
6. Reinforcement Learning
7. Model Uncertainty

Introduction to TensorFlow

tf.keras

1. High-level submodule for neural networks.
2. Sequential model, functional model, custom (subclassing).
3. Provides TF-related integration not included in standalone Keras.

tf.estimator

1. Tree-based models, linear models, neural networks.
2. Restricted framework with small number of choices.
3. Eliminates common errors and is ideal for production settings.

Introduction to TensorFlow

TensorFlow 1

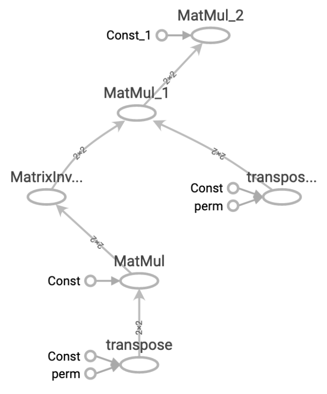
```
>>> import tensorflow as tf
>>> c = tf.constant(1.0)
>>> print(c)

Tensor("Const_2:0",
shape=(), dtype=float32)

>>> with tf.Session() as sess:
        print(c.eval())

1.0
```

TensorFlow 2

```
>>> import tensorflow as tf
>>> c = tf.constant(1.0)
>>> print(c)

<tf.Tensor: shape=(),
dtype=float32, numpy=1.0>
```

Introduction to TensorFlow

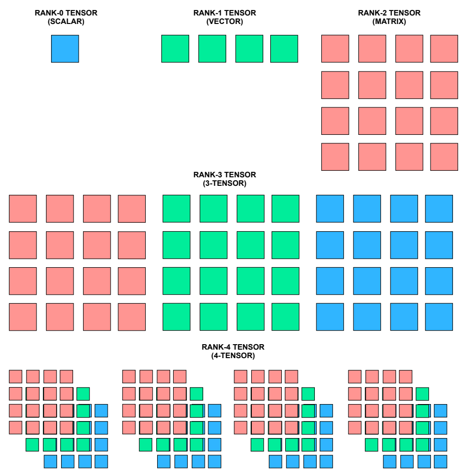
Static Graph



TensorFlow 2

```
>>> @tf.function
>>> def ols_predict(X, beta):
        yhat = tf.matmul(X, beta)
        return yhat
```

Introduction to TensorFlow



Source: “Machine Learning for Economics and Finance in TensorFlow 2”

Introduction to TensorFlow

Tensor Definitions

>>> scalar = tf.constant(1., tf.float32)

>>> matrix = tf.Variable([[1., 2.], [3., 4.]], tf.float32)

>>> tensor = tf.random.normal((2, 4, 6, 3))

Operation Definitions

Operation	Example
tf.add()	tf.add(scalar, tensor)
tf.multiply()	tf.multiply(scalar, matrix)
tf.matmul()	tf.matmul(matrix, matrix)

Automatic Differentation

- Compute $\partial g(f(x))/\partial x$.
  1. $g(y) = 3y$
  2. $f(x) = x^2$
  3. $x = 2$

Machine Learning in Economics and Finance

Symbolic

1. $g(f(x)) = 3x^2$
2. $\frac{\partial g(f(x))}{\partial x} = 6x$
3. $\frac{\partial g(f(x))}{\partial x}_{|x=2} = 12$

Numerical

1. $g(f(x)) = 3x^2$
2. $\frac{\partial g(f(x))}{\partial x} \approx \frac{3(x+h)^2 - 3x^2}{h}$
3. $\frac{\partial g(f(x))}{\partial x} \approx 6x + h$
4. $\frac{\partial g(f(x))}{\partial x}_{|x=2} \approx 12 + h$

Auto

1. $\frac{\partial g(f(x))}{\partial x} = \frac{\partial g(y)}{\partial y} \frac{\partial f(x)}{\partial x}$
2. $\frac{\partial g(y)}{\partial y} = 3$
3. $\frac{\partial f(x)}{\partial x} = 2x$
4. $\frac{\partial f(x)}{\partial x}_{|x=2} = 4$
5. $\frac{\partial g(y)}{\partial y}_{|x=2} = 3$
6. $\frac{\partial g(f(x))}{\partial x}_{|x=2} = 12$

Structured Data

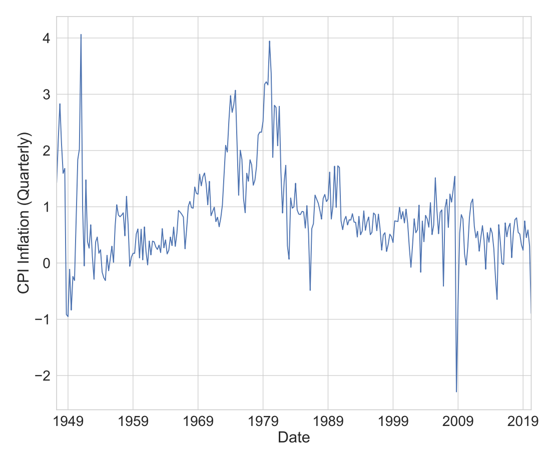
Structured Data

Data and Models in Economics

- **Structured datasets are abundant in economics and finance.**
  - Feature extraction may improve fit or predictive power, but is not a necessary first step.
- **Economic and financial models are typically linear and parsimonious.**
  - Penalized regression is common, but tree-based models and neural networks are not (yet).

Structured Data

Forecasting CPI Inflation



Source: “Machine Learning for Economics and Finance in TensorFlow 2”

Structured Data

Tabular Data

Date	Inflation	Unemployment	Hours	Earnings	M1
4/1/67	0.30	-0.44	-0.50	0.37	-0.34
...	...	...	...	...	...
12/1/19	-0.09	0.07	0.48	0.22	0.75
1/1/20	0.39	0.64	-1.67	-0.09	0.00
2/1/20	0.27	-0.23	0.00	0.45	0.82
3/1/20	-0.22	0.77	-0.24	0.36	6.44

Source: "Machine Learning for Economics and Finance in TensorFlow 2."

Structured Data

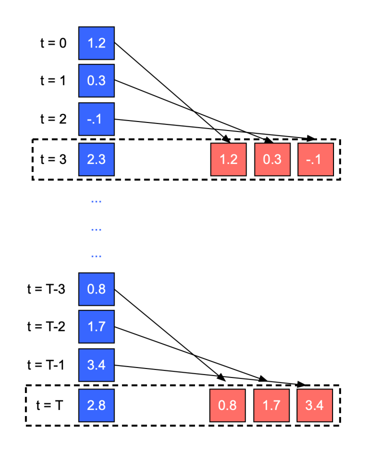
```
# Import TensorFlow and preprocessing modules.
>>> import numpy as np
>>> import pandas as pd
>>> import tensorflow as tf
>>> from tensorflow.keras.preprocessing.sequence \
        import Timeseriesgenerator
```

Structured Data

```
# Load data.
>>> macroData= pd.read_csv('macroData.csv')

# Convert to numpy array.
>>> inflation = np.array(macroData['inflation'])

# Instantiate time series generator.
>>> generator = TimeseriesGenerator(inflation, inflation,
        length = 4, batch_size = 12)
```

Structured Data

Time Series Generator



Source: "Machine Learning for Economics and Finance in TensorFlow 2"

Structured Data

```
# Define sequential model.
>>> model = tf.keras.models.Sequential()

# Add input layer.
>>> model.add(tf.keras.Input(shape=(4,)))

# Define layers.
>>> model.add(tf.keras.layers.Dense(2, activation='relu'))
>>> model.add(tf.keras.layers.Dense(1, activation='linear'))
```

Structured Data

```
# Compile the model.
>>> model.compile(loss='mse', optimizer='adam')

# Print summary of model architecture.
>>> print(model.summary())
```

Structured Data

Model: “sequential”

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 2)	10
dense_1 (Dense)	(None, 1)	3

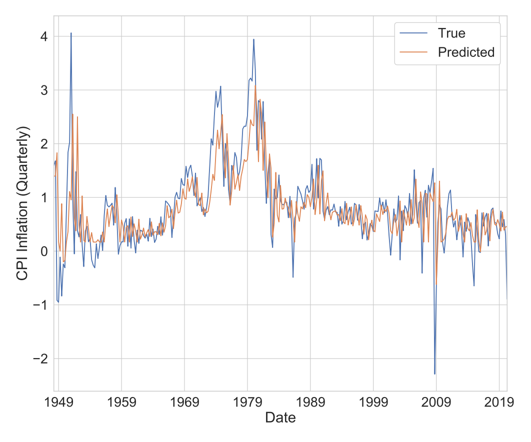
Total params: 13
Trainable params: 13
Non-trainable params: 0

Structured Data

```
# Train the model.
>>> model.fit_generator(generator, epochs=100)

Epoch 1/100
25/25 [====================] - 0s 647us/step - loss: 4.3368
...
Epoch 99/100
25/25 [====================] - 0s 658us/step - loss: 0.4504
Epoch 100/100
25/25 [====================] - 0s 650us/step - loss: 0.4467
```

Structured Data

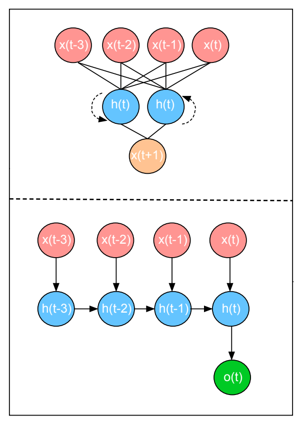
One-Quarter-Ahead Forecast (model.predict())



Source: “Machine Learning for Economics and Finance in TensorFlow 2”

Structured Data

Sequential Models



Source: “Machine Learning for Economics and Finance in TensorFlow 2”

Structured Data

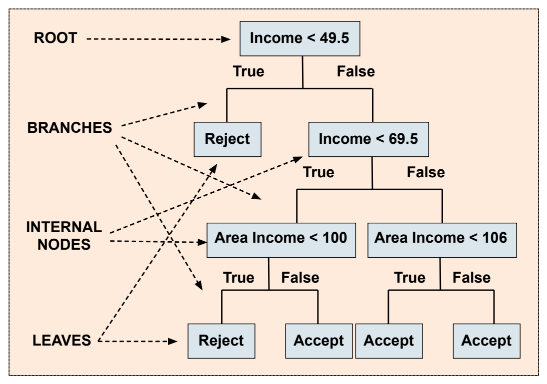
```
# Replace dense layer with LSTM.
>>> model.add(tf.keras.layers.Dense(2, activation='relu'))
>>> model.add(tf.keras.layers.LSTM(2, activation='relu'))
>>> model.add(tf.keras.layers.Dense(1, activation='linear'))
```

Tree-Based Models

1. **Perform sequential partition of data.**
   - Move from “root” to “leaves.”
2. **Achieve state-of-the-art forecasting performance on tabular data.**
   - Gradient boosted trees.
3. **Multiple implementations in TensorFlow.**
   - tf.estimator and tfdf extension.

Structured Data

Tree-Based Model: Loan Originations



Source: "Machine Learning for Economics and Finance in TensorFlow 2"

Structured Data

Gradient Boosted Trees

$$Y = G_i(X) + T_i(X) + \epsilon \tag{1}$$

$$\epsilon = Y - G_i(X) - T_i(X) \tag{2}$$

$$G_{i+1}(X) = G_i(X) + T_i(X) \tag{3}$$

Structured Data

```
# Define lagged inflation feature column.
>>> inflation = tf.feature_column.\
        numeric_column("inflation")

# Define unemployment feature column.
>>> unemployment = tf.feature_column.\
        numeric_column("unemployment")
```

Structured Data

```
# Define length of dataset.
>>> N = len(macroData)

# Define input function for training data.
>>> def train_data():
        train = macroData.iloc[:N-1]
        features = {“inflation”: train[“Inflation”],
        “unemployment”: train[“Unemployment”]}
        labels = macroData[“Inflation”].iloc[1:N]
        return features, labels
```

Structured Data

```
# Define feature list.
>>> feature_list = [inflation, unemployment]

# Define model.
>>> model = tf.estimator.BoostedTreesRegressor(
        feature_columns = feature_list,
        n_batches_per_layer = 1)

# Train model.
>>> model.train(train_data, steps = 100)
```

Structured Data

```
# Evaluate model.
>>> train_eval = model.evaluate(train_data, steps = 1)
>>> print(pd.Series(train_eval))

average_loss          0.010534
label/mean            0.416240
loss                  0.010534
prediction/mean       0.416263
global_step           100.00
dtype: float64
```

Extracting Features from Text Data

Extracting Features from Text

Text Data

- **Under-exploited source of novel and potentially useful features.**
  - Newspaper articles, social media content, central bank announcements, earnings calls, financial filings.
- **Text is unstructured and must be converted to numerical format before inclusion in model.**
  - Need a mapping from raw text, D, to numerical array, C.
- **See "Text as Data" (Gentzkow et al., 2019) for overview of theory.**

Extracting Features from Text

Text Data

- **What is D?**
  - Document corpus: $\{D_0, ..., D_{n-1}\}$.
- **What is C?**
  - Language tokens.

Extracting Features from Text

Text Features

1. Word Counts
2. Sentiment Analysis
3. Economic Policy Uncertainty
4. Topic Proportions
5. Entropy
6. Memory
7. Transfer Learning
8. Embeddings

Extracting Features from Text

Sentiment Analysis

Positivity: $\frac{2}{135}$, Negativity: $\frac{2}{135}$, Net Positivity: $0 = \frac{2}{135} - \frac{2}{135}$

Economic activity in Sweden remains strong and inflation is close to the target of 2 per cent. Uncertainty abroad has increased but new information since the monetary policy decision in April has not led to any major revisions of the forecasts overall. With continued support from monetary policy, the conditions for inflation to remain close to the target in the period ahead are considered good. The Executive Board has decided to hold the repo rate unchanged at –0.25 per cent. The forecast for the repo rate is also unchanged and indicates that it will be increased again towards the end of the year or at the beginning of next year. However, the risks surrounding developments abroad can have a bearing on the prospects for Sweden, which emphasises the importance of proceeding cautiously with monetary policy.

Extracting Features from Text

Word Counts

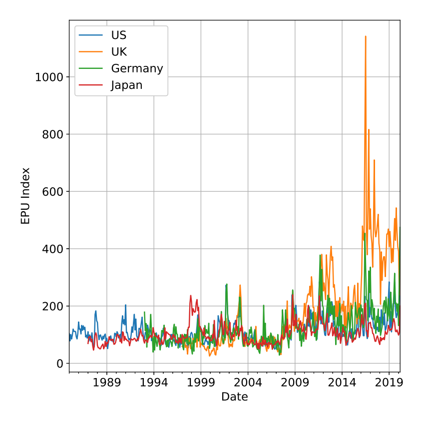
```
# Import count vectorizer.
>>> from sklearn.feature_extraction.text
        import CountVectorizer

# Instantiate vectorizer.
vectorizer = CountVectorizer(
max_features = 1000)

# Transform texts into count matrix.
C = vectorizer.fit_transform(texts)
```

Sentiment Analysis

```
# Import sentiment analysis library.
>>> import pysentiment2 as ps

# Instantiate Loughran-McDonald (2011)
dictionary.
>>> lm = ps.LM()

# Tokenize texts.
>>> tokens = [lm.tokenize(t) for
        t in texts]

# Compute sentiment scores.
>>> sentiment =
[lm.get_score(p)['Polarity'] for p
in tokens]
```

Extracting Features from Text

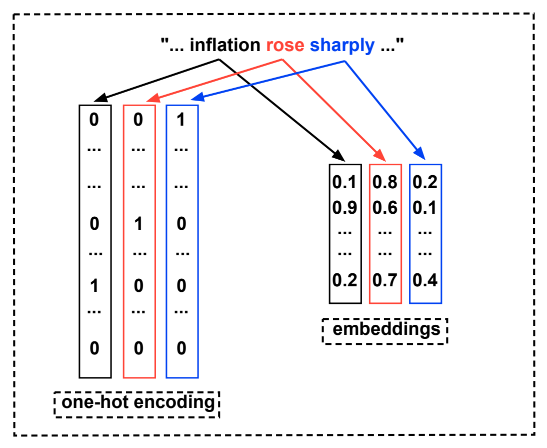
Economic Policy Uncertainty



Source: "Machine Learning for Economics and Finance in TensorFlow 2"

Extracting Features from Text

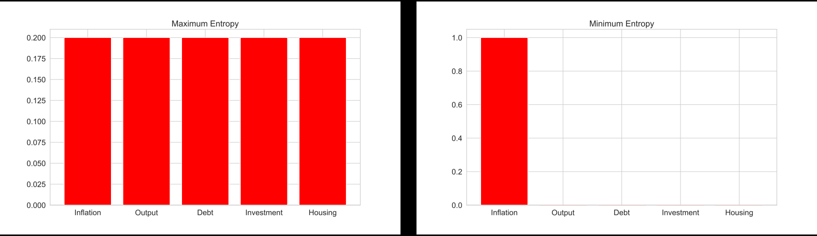
Embeddings



Source: "Machine Learning for Economics and Finance in TensorFlow 2"

Extracting Features from Text

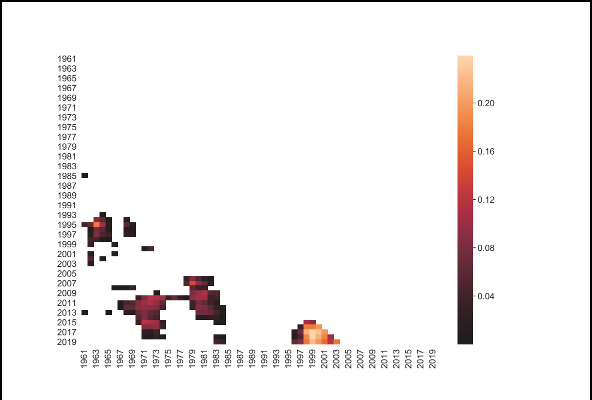
Entropy



Source: Bertsch, Hull, and Zhang (2021)

Extracting Features from Text

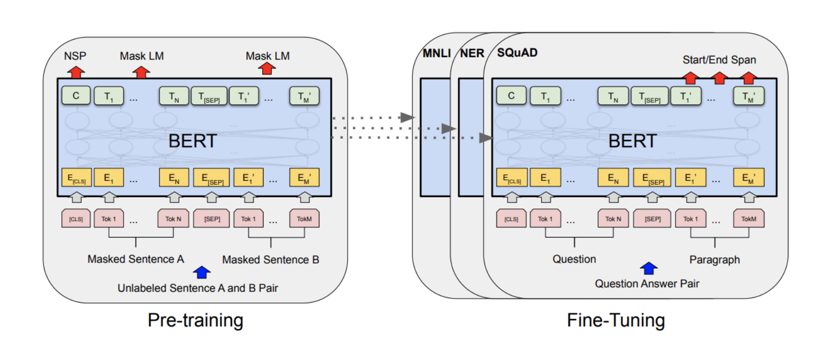
Memory



Source: Bertsch, Hull, and Zhang (2021)

Recent Developments

Feature Extraction with BERT



Source: “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Devlin et al. (2018)

Extracting Features from Text

Adding Text Features to ML Models

- **Features can be combined with tabular data or used in a standalone sequential model.**
  - LSTM model, fine-tuned transformer, multi-input NN (functional API), boosted trees.
- **Live training: text feature extraction with transformer models.**

Generative Adversarial Networks

Generative Adversarial Networks

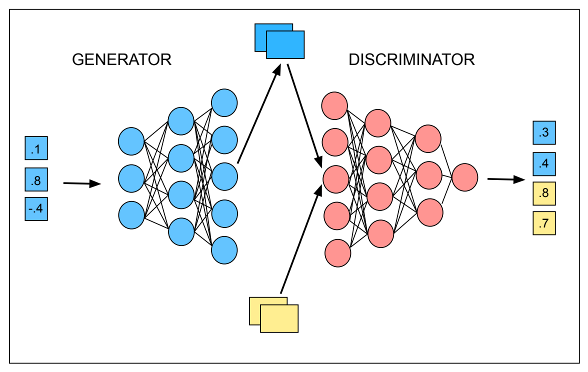
GAN Applications in Economics

1. **Generate data that appears similar to sample (Athey et al. 2019).**
   - Useful alternative to standard designs for Monte Carlo studies when available data sample is short.

2. **Estimate theoretical model using indirect inference (Kaji et al. 2018).**
   - Train model to generate data that discriminator cannot distinguish from real series.

Generative Adversarial Networks

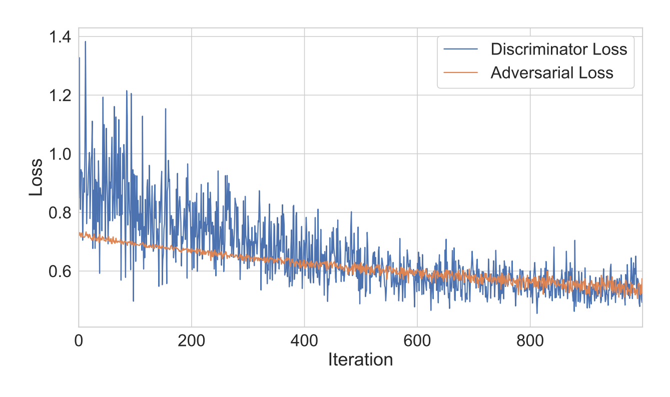
Generative Adversarial Networks (GANs)



Source: “Machine Learning for Economics and Finance in TensorFlow 2”

Generative Adversarial Networks

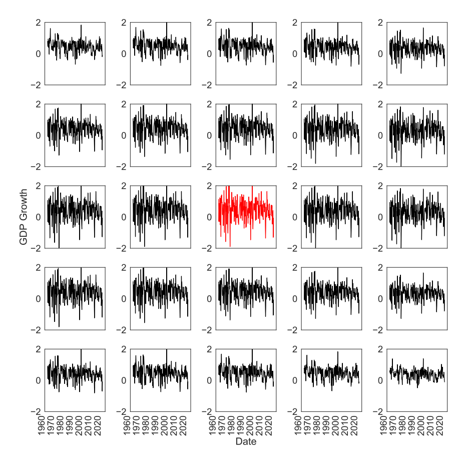
Stable Evolutionary Equilibrium



Source: “Machine Learning for Economics and Finance in TensorFlow 2”

Generative Adversarial Networks

Simulated GDP Growth Series



Source: "Machine Learning for Economics and Finance in TensorFlow 2"

Generative Adversarial Networks

<u>GANs in TensorFlow</u>

- **tf.keras simplifies construction of GANs in TensorFlow.**
  - Must define generator, discriminator, and adversarial network.
  - Share weights, but do not allow discriminator to update during generator training.
- **Live training: construct GAN to simulate GDP growth time series in TensorFlow.**