HALMSTAD UNIVERSITY

# Short Introduction of Keras

Kunru Chen

The Python part of *Introduction to research in embedded and intelligent systems*

April 20th, 2022

# About Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow.

- Simple
- Flexible
- Powerful

# About Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow.

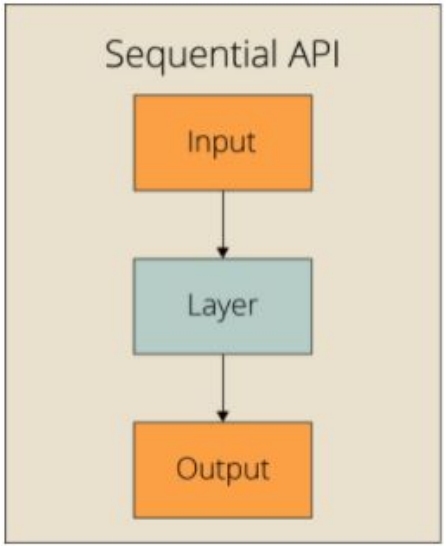
```
>>> import tensorflow as tf
>>> tf.keras.layers.cnn()
```

---

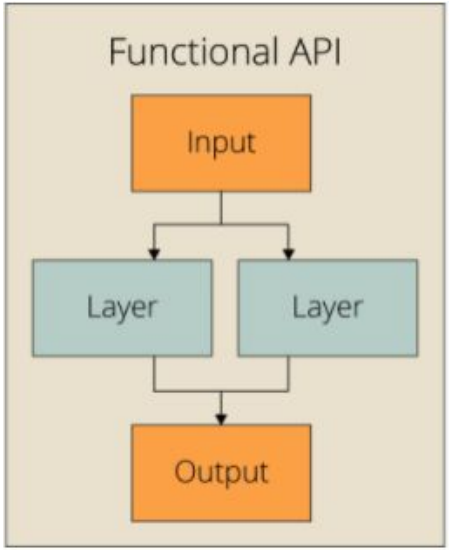
```
>>> import keras
>>> keras.layers.cnn()
```

# Define models

## Sequential API (outdated?)

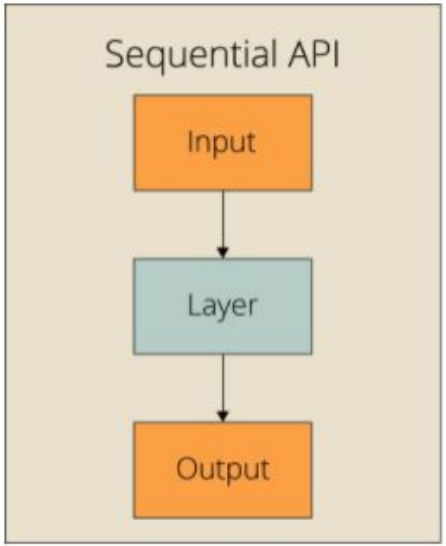



## Functional API (recommended)



# Define models

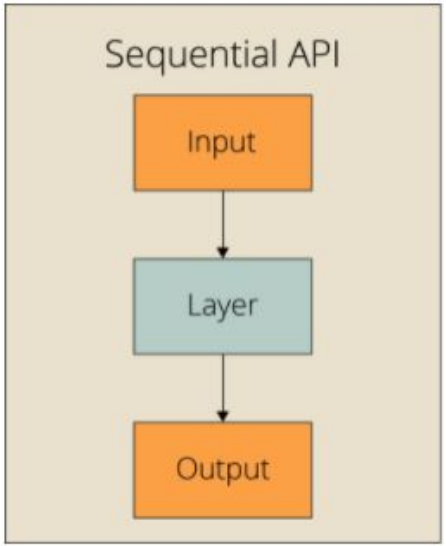
## Sequential API (outdated?)




```
>>> # Construction
... model = tf.keras.models.Sequential()
... model.add(tf.keras.layers.Dense(
...     128, input_shape=(784,), activation="tanh"))
... model.add(tf.keras.layers.Dense(
...     32, activation="tanh"))
... model.add(tf.keras.layers.Dense(
...     2, activation="softmax"))
...
... # Compilation
... model.compile(tf.keras.optimizers.Adam(),
...               tf.keras.losses.BinaryCrossentropy())
...
```

# Define models

## Sequential API (outdated?)

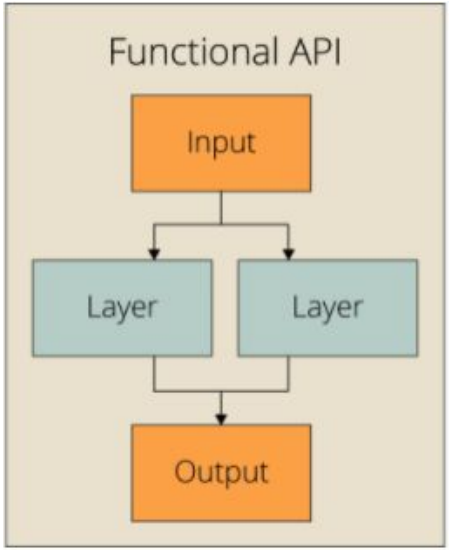



```
>>> # Define Sequential model with 3 layers
... model = tf.keras.Sequential(
...     [
...         tf.keras.layers.Dense(128, activation="tanh"),
...         tf.keras.layers.Dense(32, activation="tanh"),
...         tf.keras.layers.Dense(2, activation="softmax"),
...     ]
... )
... # Call model on a test input
... temp = model(tf.ones((1000, 784)))
...
```

# Define models (API)

```
>>> # Functional API
... _input = tf.keras.layers.Input(shape=(784,))
... x = tf.keras.layers.Dense(128, activation="relu")(_input)
... x = tf.keras.layers.Dense(32, activation="relu")(x)
... _output = tf.keras.layers.Dense(2, activation="relu")(x)
... model = tf.keras.models.Model(inputs=_input, outputs=_output)
...
... # Compilation
... model.compile(tf.keras.optimizers.Adam(),
...               tf.keras.losses.BinaryCrossentropy())
...
```

## Functional API (recommended)



# Layers

- Unit / Neuron types
  - Dense layers
  - CNN layers
  - RNN layers
- ~~Layers~~ → Techniques
  - Activation layer
  - Reshape layer
  - Timedistributed layer

# Layers

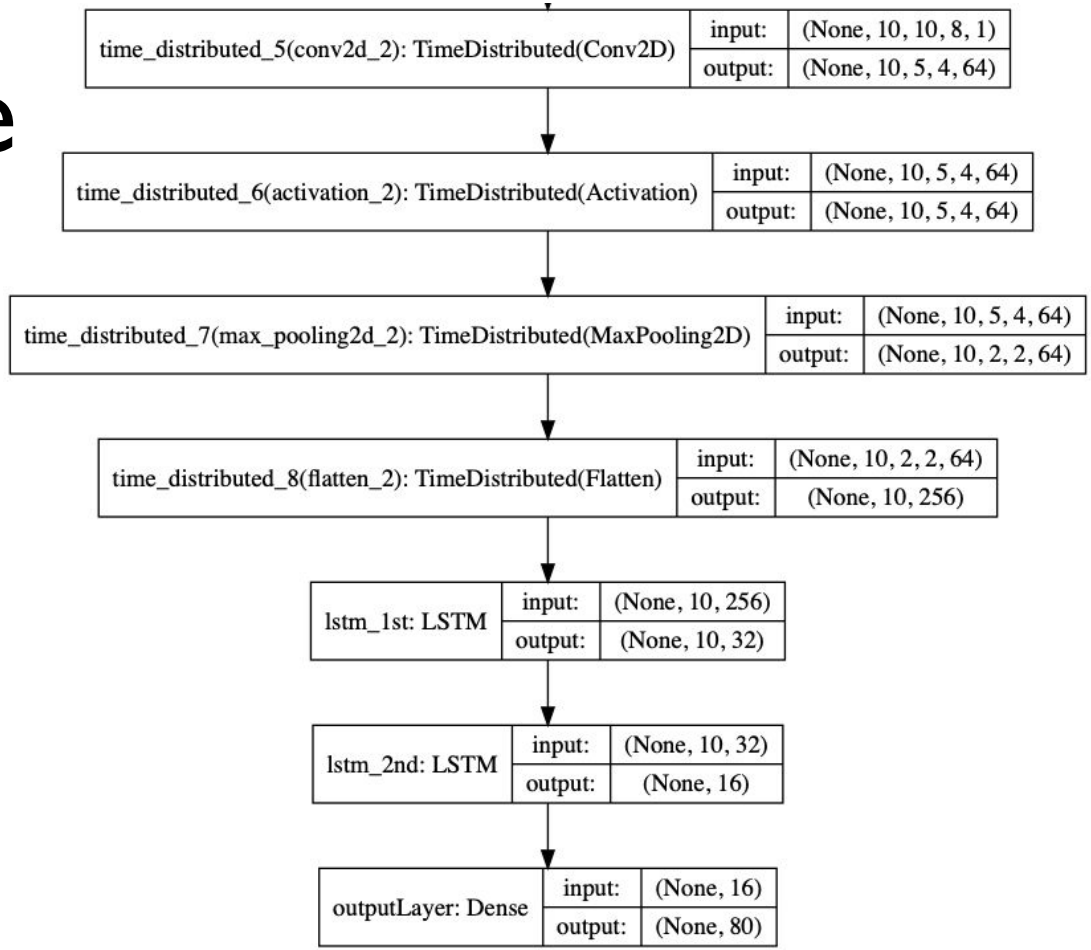
- Unit / Neuron types
  - Dense layers
  - CNN layers
  - RNN layers
- ~~Layers~~ → Techniques
  - Activation layer
  - Reshape layer
  - Timedistributed layer

## Layers API

- The base Layer class
- Layer activations
- Layer weight initializers
- Layer weight regularizers
- Layer weight constraints
- Core layers
- Convolution layers
- Pooling layers
- Recurrent layers
- Preprocessing layers
- Normalization layers
- Regularization layers
- Attention layers
- Reshaping layers
- Merging layers
- Locally-connected layers
- Activation layers

# Layers Example

- Layer-type
- Input size
  - 10 windows
  - 10 timesteps
  - 8 features
  - 1 dimension



# Compile

Compile defines the loss function, the optimizer and the metrics.

- Optimizers
- Losses

```
>>> # Construction
... model = tf.keras.models.Sequential()
... model.add(tf.keras.layers.Dense(
...     128, input_shape=(784,), activation="tanh"))
... model.add(tf.keras.layers.Dense(
...     32, activation="tanh"))
... model.add(tf.keras.layers.Dense(
...     2, activation="softmax"))
...
... # Compilation
... model.compile(tf.keras.optimizers.Adam(),
...               tf.keras.losses.BinaryCrossentropy())
...
```

# Compile

Compile defines the loss function, the optimizer and the metrics.

- Optimizers
- Losses

**Optimizers**

- SGD
- RMSprop
- Adam
- Adadelta
- Adagrad
- Adamax
- Nadam
- Ftrl

# Compile

Compile defines the loss function, the optimizer and the metrics.

- Optimizers
- Losses

**Losses**

- Probabilistic losses
- Regression losses
- Hinge losses for "maximum-margin" classification

# Compile

Compile defines the loss function, the optimizer and the metrics.

- Optimizers
- Losses

### Losses

- Probabilistic losses
- Regression losses
- Hinge losses for "maximum-margin" classification

### Probabilistic losses

- `BinaryCrossentropy` class
- `CategoricalCrossentropy` class
- `SparseCategoricalCrossentropy` class
- `Poisson` class
- `binary_crossentropy` function
- `categorical_crossentropy` function
- `sparse_categorical_crossentropy` function
- `poisson` function
- `KLDivergence` class
- `kl_divergence` function

# Callback

A callback is an object that can perform actions at various stages of training.

- Checkpoint
- EarlyStopping
- TensorBoard

```
>>> my_callbacks = [
...     tf.keras.callbacks.EarlyStopping(patience=2),
...     tf.keras.callbacks.ModelCheckpoint(
...         filepath='model.{epoch:02d}-{val_loss:.2f}.h5'),
...     tf.keras.callbacks.TensorBoard(log_dir='./logs'),
... ]
... model.fit(dataset, callbacks=my_callbacks)
...
```

# Callback

A callback is an object that can perform actions at various stages of training.

- Checkpoint
- EarlyStopping
- TensorBoard

## Available callbacks

- Base Callback class
- ModelCheckpoint
- TensorBoard
- EarlyStopping
- LearningRateScheduler
- ReduceLROnPlateau
- RemoteMonitor
- LambdaCallback
- TerminateOnNaN
- CSVLogger
- ProgbarLogger
- BackupAndRestore

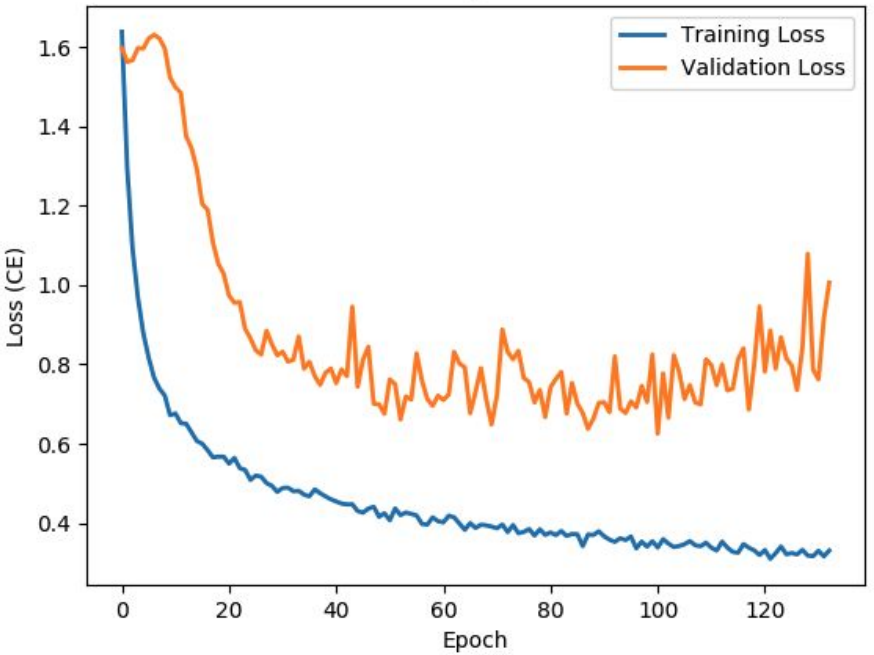
# Training

```
>>> checkpoint = ModelCheckpoint("/Users/kunche/Desktop/testing.h5",
...                              monitor='val_loss',
...                              verbose=0,
...                              save_best_only=True,
...                              mode='min')
... stop = EarlyStopping(monitor='val_loss', patience=48, verbose=1)
... callbacks_list = [checkpoint, stop]
... model = building_mlp()
... h = model.fit(X_train, y_train,
...               validation_split=0.2,
...               epochs=200,
...               shuffle=True,
...               batch_size=10,
...               callbacks=callbacks_list,
...               verbose=2)
... model.load_weights("/Users/kunche/Desktop/testing.h5")
... model.save("/Users/kunche/Desktop/modelsTemp/exampleModel.h5")
...
```

```
Model.fit(
    x=None,
    y=None,
    batch_size=None,
    epochs=1,
    verbose="auto",
    callbacks=None,
    validation_split=0.0,
    validation_data=None,
    shuffle=True,
    class_weight=None,
    sample_weight=None,
    initial_epoch=0,
    steps_per_epoch=None,
    validation_steps=None,
    validation_batch_size=None,
    validation_freq=1,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False,
)
```

# After Training

```
>>> h = model.fit(x, y)
>>> h.history["loss"]
>>> h.history["val_loss"]
>>> plt.figure…
```

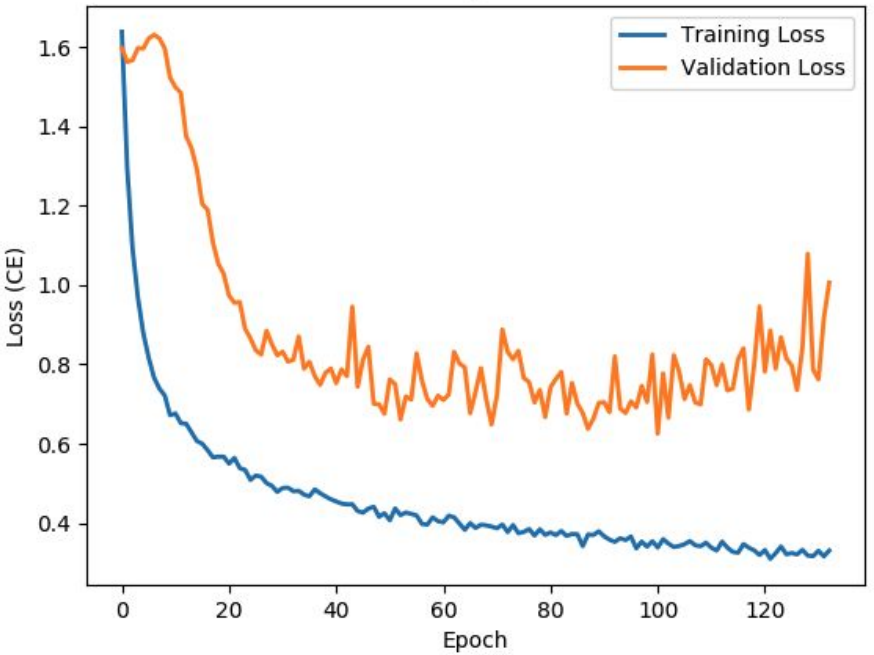


# After Training

```
>>> h = model.fit(x, y)
>>> h.history["loss"]
>>> h.history["val_loss"]
>>> plt.figure…
```

---

```
>>> model.evaluate(x, y)
```



# ~~Preprocessing~~ Preparation

## Available dataset preprocessing utilities

### Image data preprocessing

- image_dataset_from_directory function
- load_img function
- img_to_array function

### Timeseries data preprocessing

- timeseries_dataset_from_array function
- pad_sequences function

### Text data preprocessing

- text_dataset_from_directory function

```
main_directory/
...class_a/
......a_image_1.jpg
......a_image_2.jpg
...class_b/
......b_image_1.jpg
......b_image_2.jpg
```

# Preprocessing

- Text
  - tf.keras.layers.TextVectorization
- Numerical
  - tf.keras.layers.Normalization
  - tf.keras.layers.Discretization
- Categorial
  - tf.keras.layers.CategoryEncoding
  - tf.keras.layers.Hashing
  - tf.keras.layers.StringLookup
  - tf.keras.layers.IntegerLookup

# Preprocessing

- Text
  - tf.keras.layers.TextVectorization
- Numerical
  - tf.keras.layers.Normalization
  - tf.keras.layers.Discretization
- Categorial
  - tf.keras.layers.CategoryEncoding
  - tf.keras.layers.Hashing
  - tf.keras.layers.StringLookup
  - tf.keras.layers.IntegerLookup
- Image
  - tf.keras.layers.Resizing
  - tf.keras.layers.Rescaling
  - tf.keras.layers.CenterCrop
  - tf.keras.layers.RandomCrop
  - tf.keras.layers.RandomFilp
  - tf.keras.layers.RandomRotation
  - tf.keras.layers.RandomZoom
  - tf.keras.layers.RandomContrast
  - ……

# Preprocessing

```
>>> inputs = tf.keras.layers.Input(shape=input_shape)
>>> x = preprocessing_layer(inputs)
>>> outputs = rest_of_the_model(x)
>>> model = keras.Model(inputs, outputs)
```

# Preprocessing

```
>>> data = [
...     "ξεῖν', ἦ τοι μὲν ὄνειροι ἀμήχανοι ἀκριτόμυθοι",
...     "γίγνοντ', οὐδέ τι πάντα τελείεται ἀνθρώποισι.",
...     "δοιαὶ γάρ τε πύλαι ἀμενηνῶν εἰσὶν ὀνείρων:",
...     "αἱ μὲν γὰρ κεράεσσι τετεύχαται, αἱ δ' ἐλέφαντι:",
...     "τῶν οἳ μέν κ' ἔλθωσι διὰ πριστοῦ ἐλέφαντος,",
...     "οἳ ῥ' ἐλεφαίρονται, ἔπε' ἀκράαντα φέροντες:",
...     "οἱ δὲ διὰ ξεστῶν κεράων ἔλθωσι θύραζε,",
...     "οἳ ῥ' ἔτυμα κραίνουσι, βροτῶν ὅτε κέν τις ἴδηται.",
... ]
... layer = tf.keras.layers.TextVectorization()
... tf.keras.layer.adapt(data)
... vectorized_text = layer(data)
... print(vectorized_text)
...
...
```

# Preprocessing

```
>>> data = [
...     "ξεῖν', ἦ τοι μὲν ὄνειροι ἀμήχανοι ἀκριτόμυθοι",
...     "γίγνοντ', οὐδέ τι πάντα τελείεται ἀνθρώποισι.",
...     "δοιαὶ γάρ τε πύλαι ἀμενηνῶν εἰσὶν ὀνείρων:",
...     "αἱ μὲν γὰρ κεράεσσι τετεύχαται, αἱ δ' ἐλέφαντι:",
...     "τῶν οἳ μέν κ' ἔλθωσι διὰ πριστοῦ ἐλέφαντος,",
...     "οἳ ῥ' ἐλεφαίρονται, ἔπε' ἀκράαντα
...     "οἱ δὲ διὰ ξεστῶν κεράων ἔλθωσι θύρ
...     "οἳ ῥ' ἔτυμα κραίνουσι, βροτῶν ὅτε
... ]
... layer = tf.keras.layers.TextVectorizati
... tf.keras.layer.adapt(data)
... vectorized_text = layer(data)
... print(vectorized_text)
...
...
```

```
tf.Tensor(
[[37 12 25  5  9 20 21  0  0]
 [51 34 27 33 29 18  0  0  0]
 [49 52 30 31 19 46 10  0  0]
 [ 7  5 50 43 28  7 47 17  0]
 [24 35 39 40  3  6 32 16  0]
 [ 4  2 15 14 22 23  0  0  0]
 [36 48  6 38 42  3 45  0  0]
 [ 4  2 13 41 53  8 44 26 11]], shape=(8, 9), dtype=int64)
```

# Application × Utils

## Utilities

### Model plotting utilities

- plot_model function
- model_to_dot function

### Serialization utilities

- custom_object_scope class
- get_custom_objects function
- register_keras_serializable function
- serialize_keras_object function
- deserialize_keras_object function

### Python & NumPy utilities

- set_random_seed function
- to_categorical function
- normalize function
- get_file function
- Progbar class
- Sequence class

### Backend utilities

- clear_session function
- floatx function
- set_floatx function
- image_data_format function
- set_image_data_format function
- epsilon function
- set_epsilon function
- is_keras_tensor function
- get_uid function
- rnn function

# Application × Utils

## Keras Applications

- Xception
- EfficientNet B0 to B7
- EfficientNetV2 B0 to B3 and S, M, L
- VGG16 and VGG19
- ResNet and ResNetV2
- MobileNet, MobileNetV2, and MobileNetV3
- DenseNet
- NasNetLarge and NasNetMobile
- InceptionV3
- InceptionResNetV2

## Utilities

### Model plotting utilities

- plot_model function
- model_to_dot function

### Serialization utilities

- custom_object_scope class
- get_custom_objects function
- register_keras_serializable function
- serialize_keras_object function
- deserialize_keras_object function

### Python & NumPy utilities

- set_random_seed function
- to_categorical function
- normalize function
- get_file function
- Progbar class
- Sequence class

### Backend utilities

- clear_session function
- floatx function
- set_floatx function
- image_data_format function
- set_image_data_format function
- epsilon function
- set_epsilon function
- is_keras_tensor function
- get_uid function
- rnn function

# Application × Utils

```
>>> from tensorflow import keras
...
... train_ds = keras.utils.image_dataset_from_directory(
...     directory='training_data/',
...     labels='inferred',
...     label_mode='categorical',
...     batch_size=32,
...     image_size=(256, 256))
... validation_ds = keras.utils.image_dataset_from_directory(
...     directory='validation_data/',
...     labels='inferred',
...     label_mode='categorical',
...     batch_size=32,
...     image_size=(256, 256))
...
... model = keras.applications.Xception(
...     weights=None, input_shape=(256, 256, 3), classes=10)
... model.compile(optimizer='rmsprop', loss='categorical_crossentropy')
... model.fit(train_ds, epochs=10, validation_data=validation_ds)
...
```

B3 and S, M, L

2

tV2, and MobileNetV3

sNetMobile

* Xception: Deep Learning with Depthwise Separable Convolutions (CVPR 2017)

## Utilities

### Model plotting utilities

- plot_model function
- model_to_dot function

### Serialization utilities

- custom_object_scope class
- get_custom_objects function
- register_keras_serializable function
- serialize_keras_object function
- deserialize_keras_object function

### Python & NumPy utilities

- set_random_seed function
- to_categorical function
- normalize function
- get_file function
- Progbar class
- Sequence class

### Backend utilities

- clear_session function
- floatx function
- set_floatx function
- image_data_format function
- set_image_data_format function
- epsilon function
- set_epsilon function
- is_keras_tensor function
- get_uid function
- rnn function

# Toward TensorFlow

## Training with multiple GPUs

```
>>> # Create a MirroredStrategy.
... strategy = tf.distribute.MirroredStrategy()
... # Open a strategy scope.
... with strategy.scope():
...   model = Model(...)
...   model.compile(...)
...
... # Train the model on all available devices.
... train_dataset, val_dataset, test_dataset = get_dataset()
... model.fit(train_dataset, epochs=2, validation_data=val_dataset)
... # Test the model on all available devices.
... model.evaluate(test_dataset)
...
```

# Others (what's missing)

- Multiple inputs / multiple outputs
- Generative models in Keras
- train_on_batch()
  - Adversarial training in Keras
- Stepping back to TensorFlow
  - Customize your own loss / layer
- Compare to other deep learning package