ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

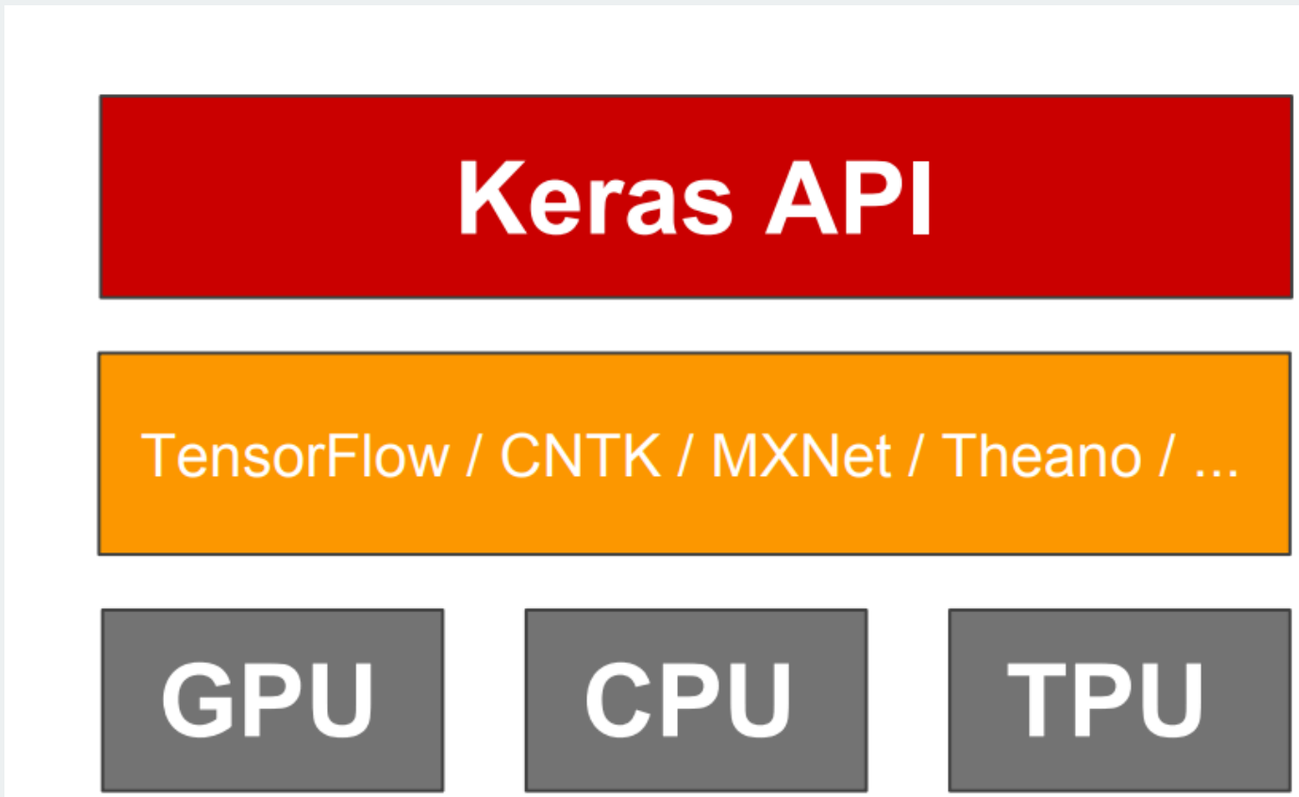
Keras

Дмитрий Музалевский
Преподаватель

План на сегодня

1. **Keras. Что и почему?**
2. Как использовать Keras
3. Распределенные вычисления
4. Практика: нейросеть на Keras

Keras API



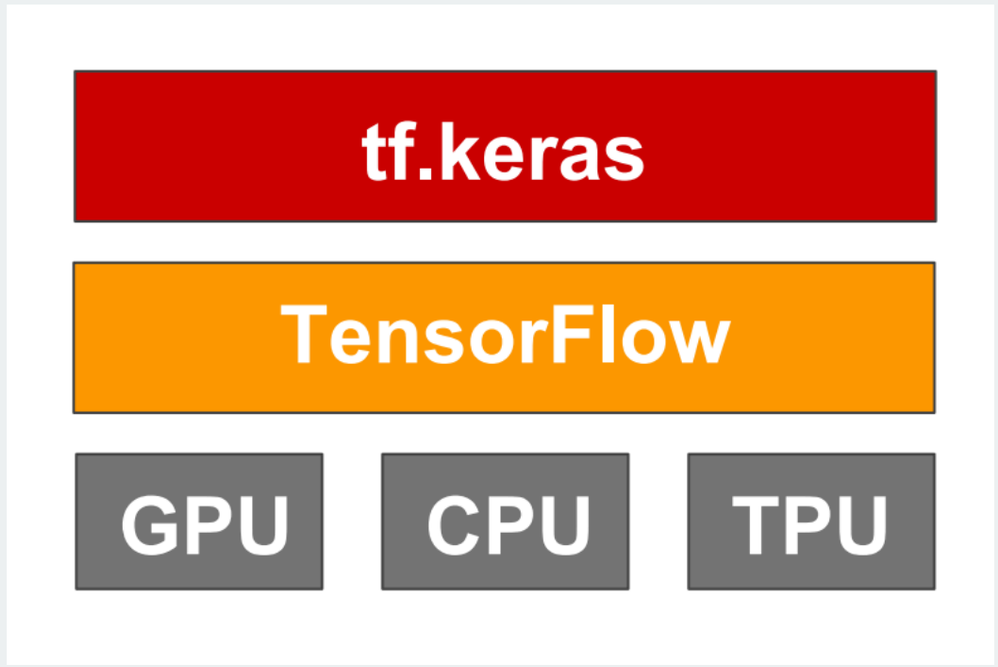
Дмитрий Музалевский, преподаватель

Keras как API для Tensorflow

1. Модуль tensorflow.keras (tf.keras)
2. Часть Tensorflow core с версии 1.4
3. Keras API
4. Оптимизация под Tensorflow
5. Интеграция с фичами Tensorflow:

   -Estimator API

   -Eager Execution, И так далее.

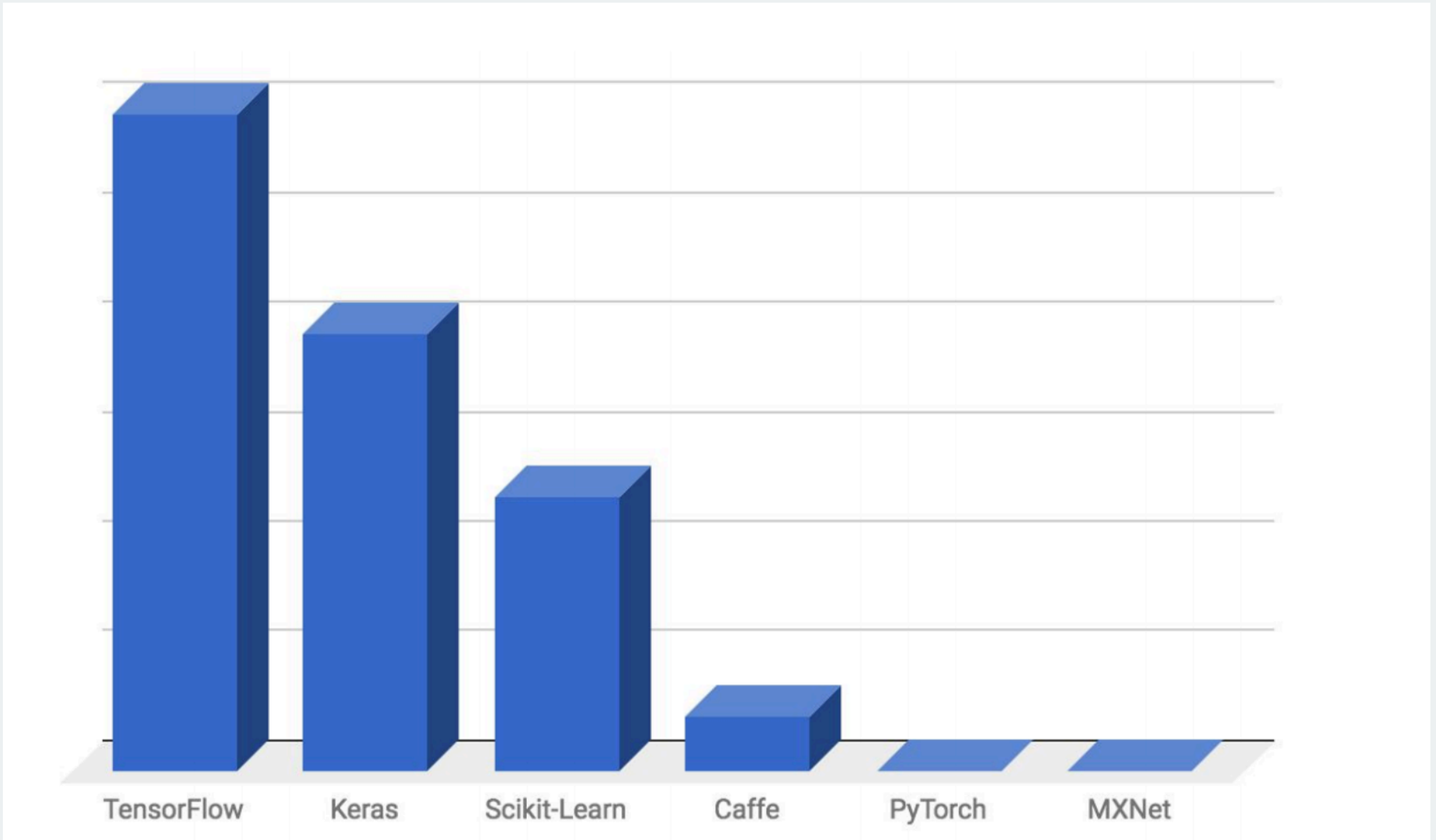


Особенности Keras

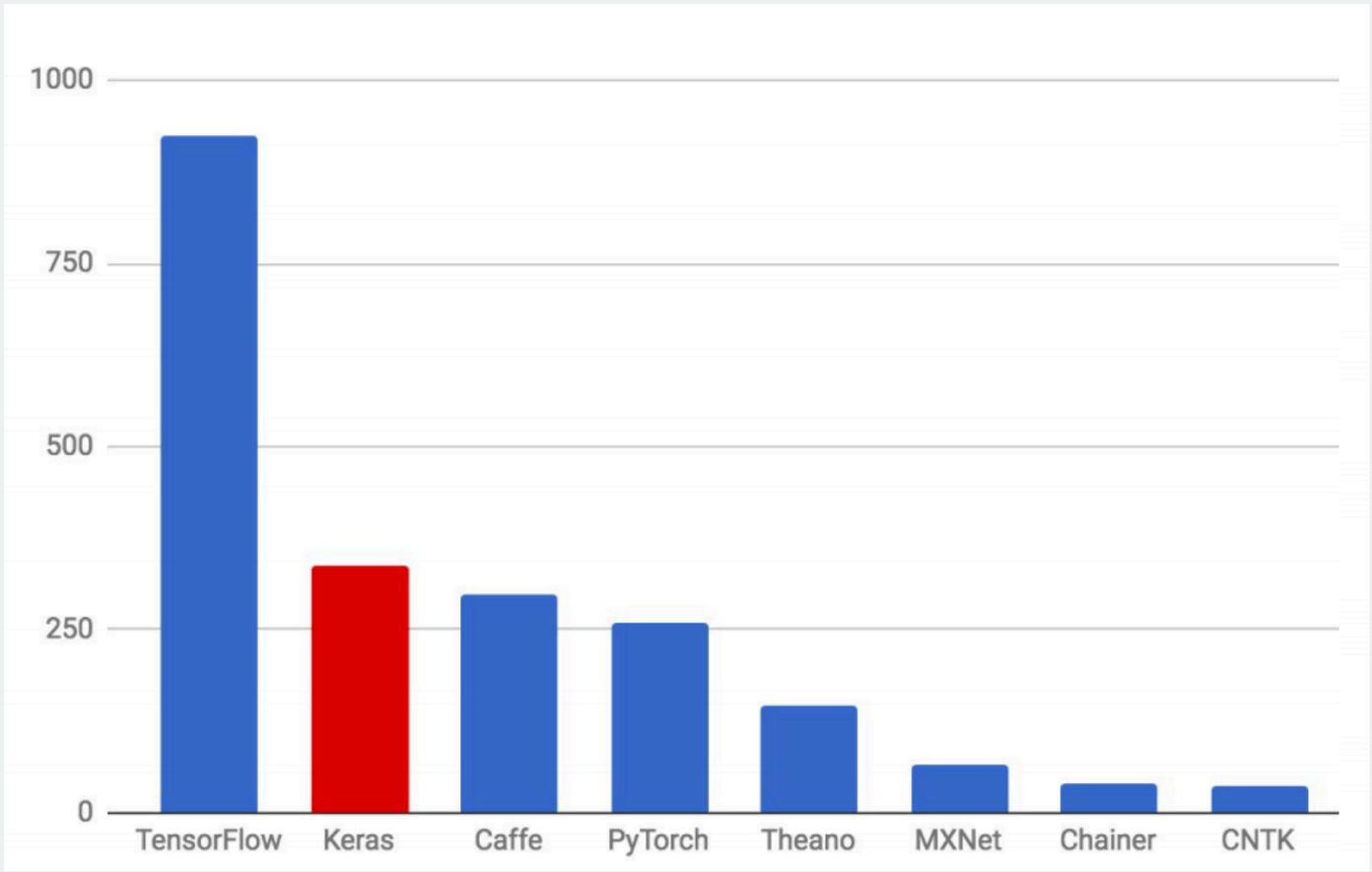
1. Фокус на User Experience
2. Широкое распространение и research community
3. Мульти-бэкэнд, мультиплатформенность
4. Сравнительно легкая возможность деплоя моделей

Использование в индустрии

Статистика по стартапам (HackerNews)



Статистика по исследованиям (arXiv)



Дмитрий Музалевский, преподаватель

1. Keras предлагает последовательные и простые API, сводит к минимуму количество действий пользователя и обеспечивает четкую и действенную обратную связь с пользователем.
2. Это делает Keras простым и эффективным в использовании. Можно быстро писать алгоритмы и модели.
3. Простота в использовании не сводится к меньшему функционалу и снижению гибкости. Это достигается использованием низкоуровневых deep learning фреймворков (tensorflow, theano).

MultiBackend/MultiPlatform

1. Возможность разработки на Python и R под Unix, OSx и Windows системы.
2. Код совместим с Tensorflow, Theano, CNTK, MXnet, PlaidML
3. Возможность использования с различными процессорными архитектурами: CPU, GPU, TPU

Production

1. TF-serving
2. Browser с возможностью GPU акселерации (WebKeras, Keras.js)
3. Android (TF, TF Lite) / Iphone (Core ML)
4. Raspberry Pi
5. JVM
6. AR apps: Keras + TF + CoreML + ARkit

План на сегодня

1. Keras. Что и почему?
2. **Как использовать Keras**
3. Распределенные вычисления
4. Практика: нейросеть на Keras

Стили API

1. Sequential Model:

   -простота

   -для single input, single output

   -покрывает до 70% задач

2. Функциональное API:

   -блочный стиль написания

   -multi input, multi output

   -покрывает до 95% задач

3. Model Subclassing:

   -максимальная гибкость использования

Sequential API

```
import keras
from keras import layers

model = keras.Sequential()
model.add(layers.Dense(20, activation='relu', input_shape=(10,)))
model.add(layers.Dense(20, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.fit(x, y, epochs=10, batch_size=32)
```

Функциональный API

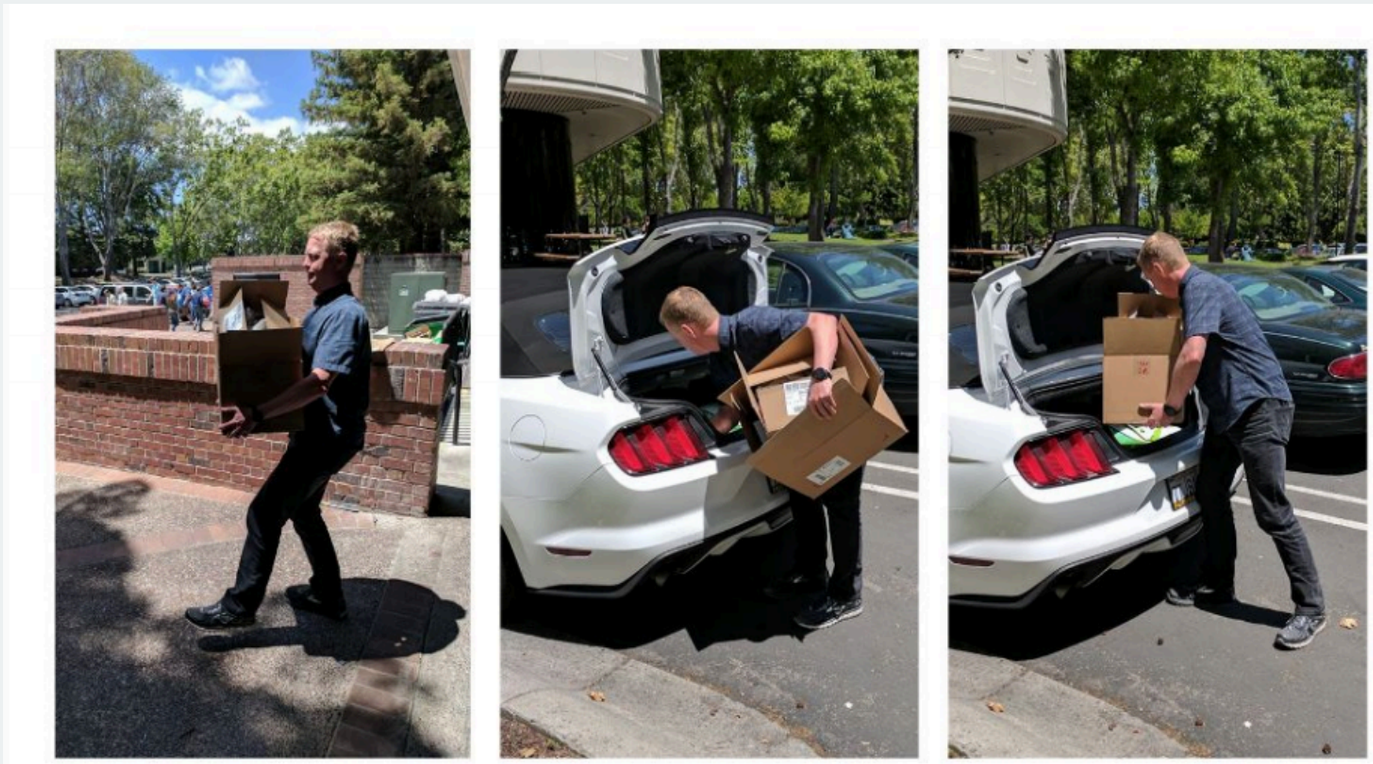
```
import keras
from keras import layers

inputs = keras.Input(shape=(10,))
x = layers.Dense(20, activation='relu')(x)
x = layers.Dense(20, activation='relu')(x)
outputs = layers.Dense(10, activation='softmax')(x)

model = keras.Model(inputs, outputs)
model.fit(x, y, epochs=10, batch_size=32)
```


Model Subclassing

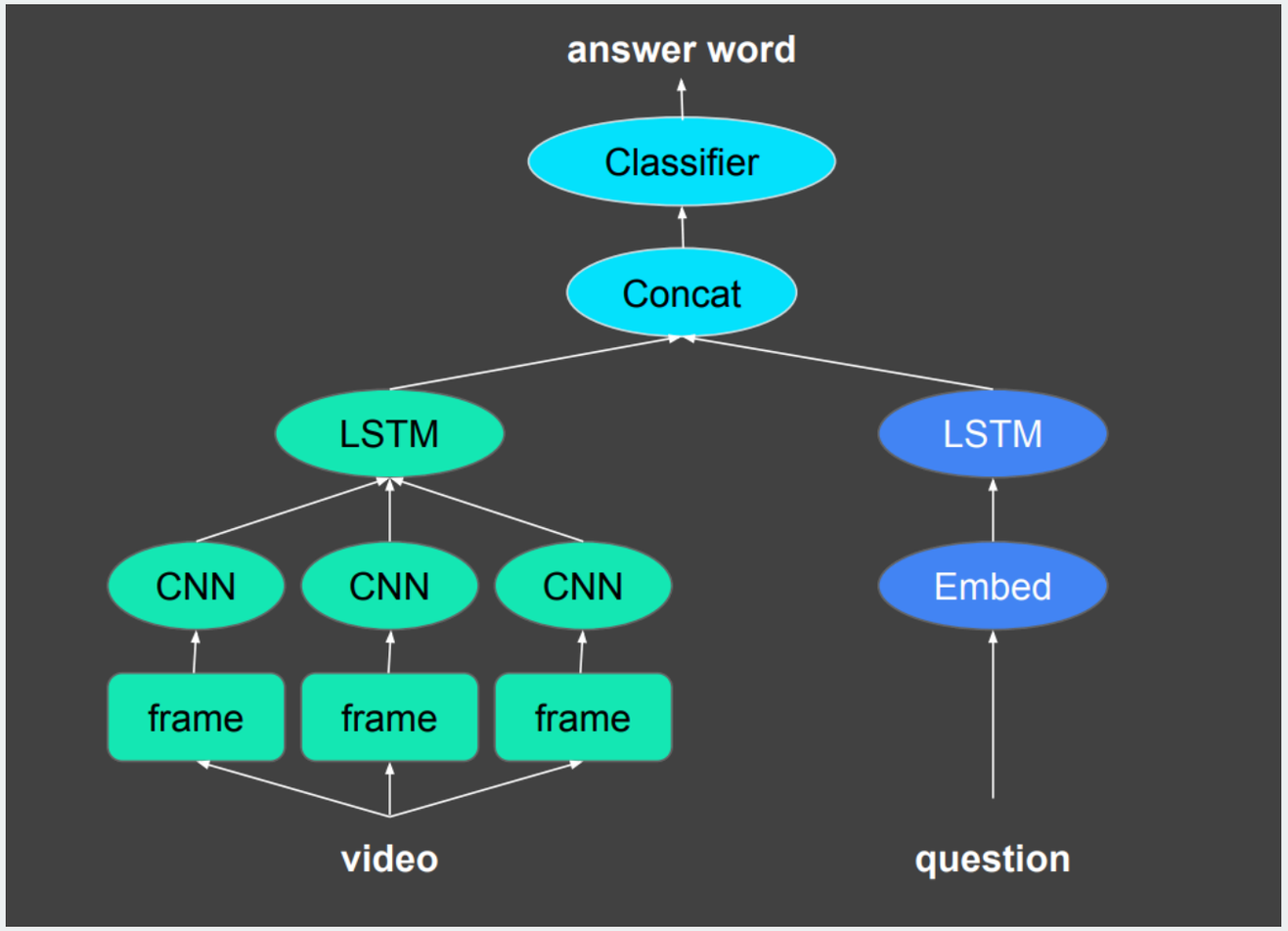
```
import keras
from keras import layers

class MyModel(keras.Model):

    def __init__(self):
        super(MyModel, self).__init__()
        self.dense1 = layers.Dense(20, activation='relu')
        self.dense2 = layers.Dense(20, activation='relu')
        self.dense3 = layers.Dense(10, activation='softmax')

    def call(self, inputs):
        x = self.dense1(x)
        x = self.dense2(x)
        return self.dense3(x)

model = MyModel()
model.fit(x, y, epochs=10, batch_size=32)
```

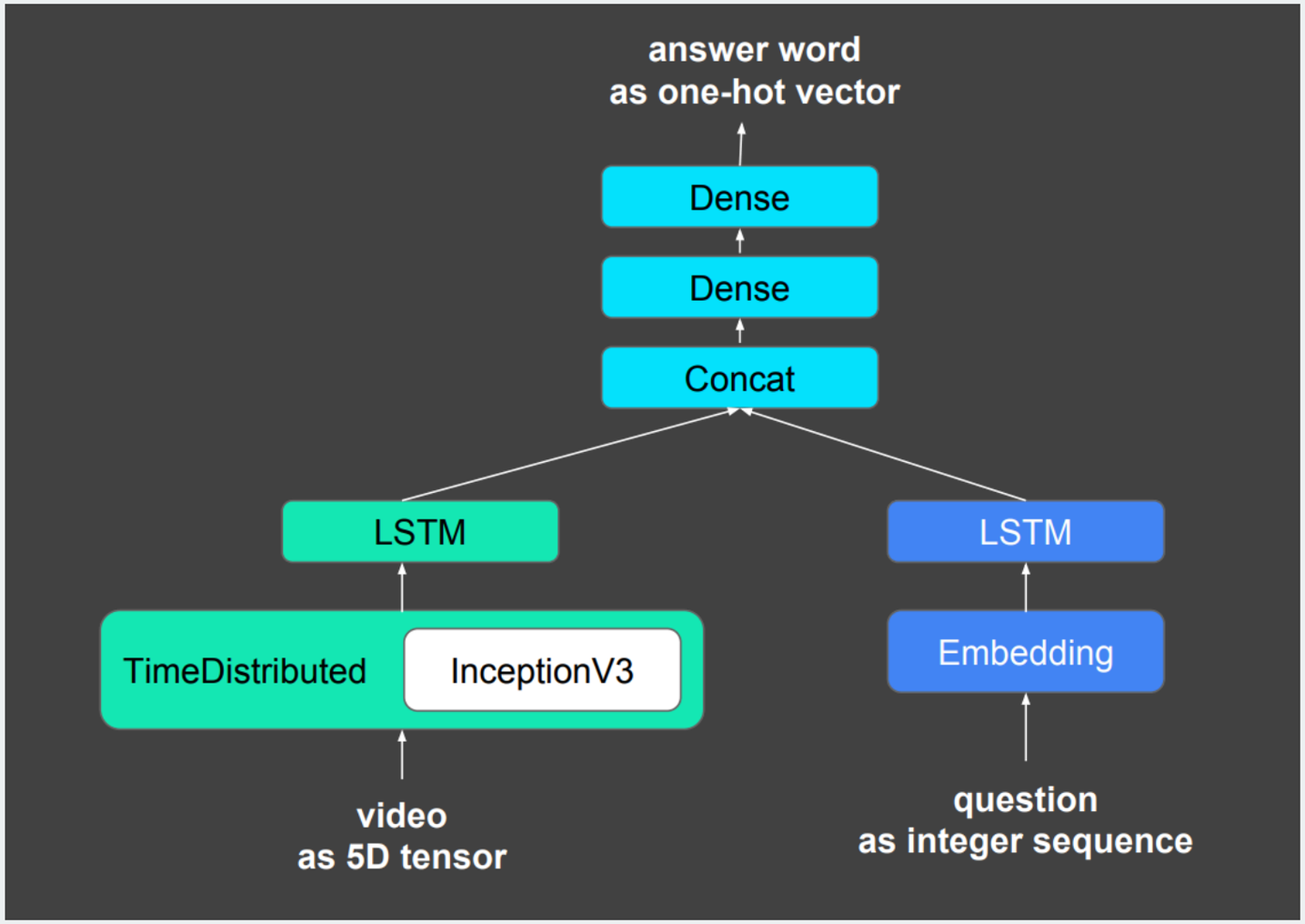

OTUS

Video Detection



Дмитрий Музалевский, преподаватель


answer word
Classifier
Concat
LSTM
LSTM
CNN
CNN
CNN
Embed
frame
frame
frame
video
question


answer word
as one-hot vector
Dense
Dense
Concat
LSTM
LSTM
TimeDistributed
InceptionV3
Embedding
video
as 5D tensor
question
as integer sequence

```python
import keras
from keras import layers
from keras.applications import InceptionV3

video = keras.Input(shape=(None, 150, 150, 3), name='video')
cnn = InceptionV3(weights='imagenet', include_top=False, pooling='avg')

cnn.trainable = False
frame_features = layers.TimeDistributed(cnn)(video)
video_vector = layers.LSTM(256)(frame_features)


question = keras.Input(shape=(None,), dtype='int32', name='question')
embedded_words = layers.Embedding(input_voc_size, 256)(question)
question_vector = layers.LSTM(128)(embedded_words)


x = layers.concatenate([video_vector, question_vector])
x = layers.Dense(128, activation=tf.nn.relu)(x)
predictions = layers.Dense(output_voc_size, activation='softmax', name='predictions')(x)


model = keras.models.Model([video, question], predictions)
model.compile(optimizer=tf.AdamOptimizer(), loss=keras.losses.categorical_crossentropy)
model.fit_generator(data_generator, steps_per_epoch=1000, epochs=100)
```

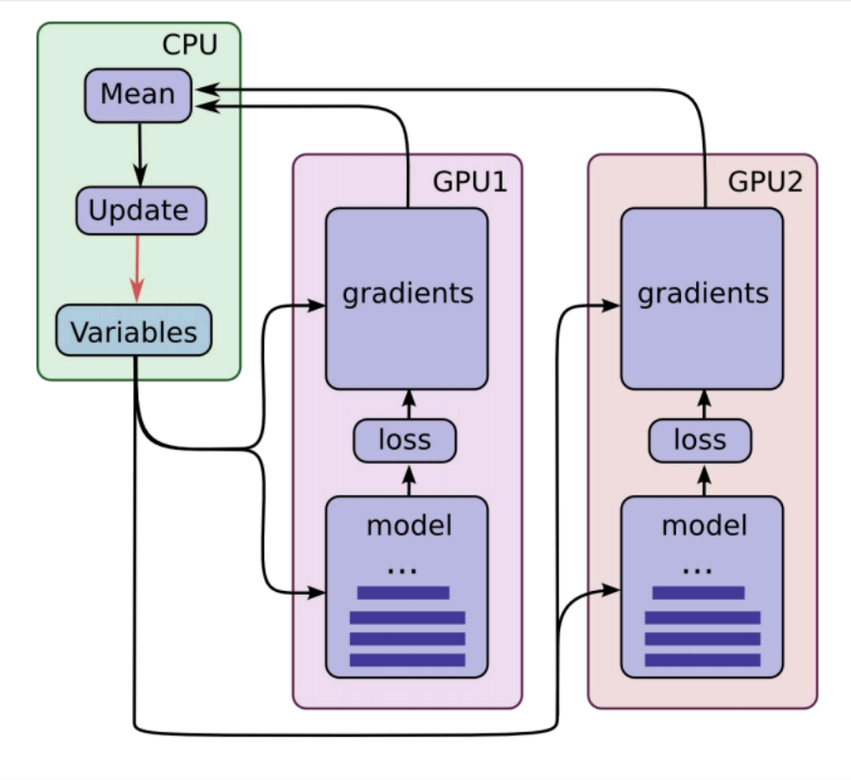

План на сегодня

1. Keras. Что и почему?
2. Как использовать Keras
3. **Распределенные вычисления**
4. Практика: нейросеть на Keras

Распределенные вычисления

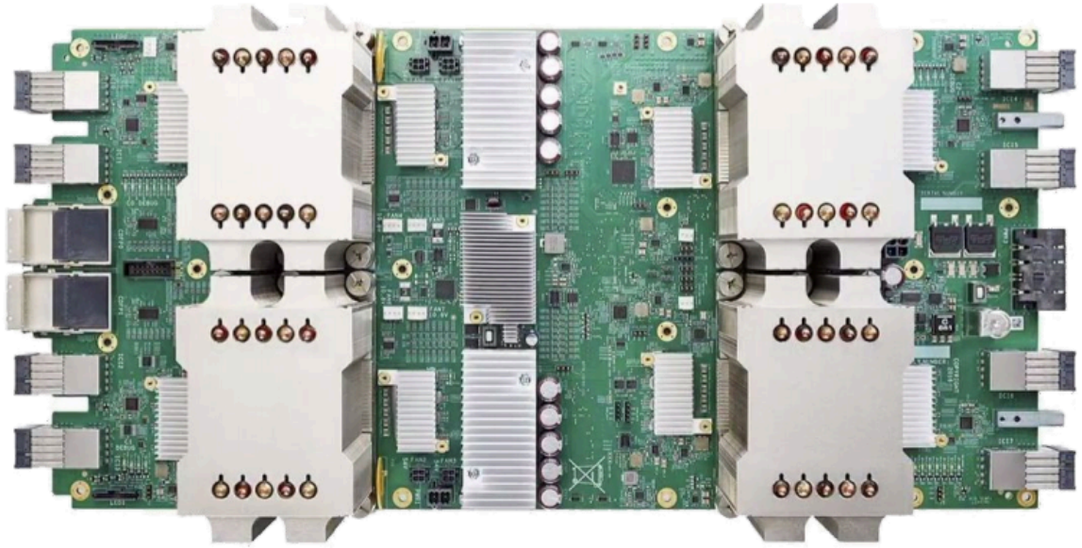
1. Estimator API (TF)
2. Horovod
3. Elephas (Spark)

GPU




```
import tensorflow as tf
from keras.applications import Xception
from keras.utils import multi_gpu_model

with tf.device('/cpu:0'):
    model = Xception(weights=None, input_shape=(height,width,3), classes=num_classes)

parallel_model = multi_gpu_model(model, gpus=8)
parallel_model.compile(loss='categorical_crossentropy', optimizer='rmsprop')

parallel_model.fit(x,y, epochs=20, batch_size=256)
```

TPU



```
import os
import tensorflow as tf

TPU_WORKER = 'grpc://' + os.environ['COLAB_TPU_ADDR']
tf.logging.set_verbosity(tf.logging.INFO)

tpu_model = tf.contrib.tpu.keras_to_tpu_model(
    training_model,
    strategy=tf.contrib.tpu.TPUDistributionStrategy(
        tf.contrib.cluster_resolver.TPUClusterResolver(TPU_WORKER)))

history = tpu_model.fit(x_train, y_train,
                        epochs=20,
                        batch_size=128 * 8,
                        validation_split=0.2)
tpu_model.save_weights('./tpu_model.h5', overwrite=True)
tpu_model.evaluate(x_test, y_test, batch_size=128 * 8)
```

План на сегодня

1. Keras. Что и почему?
2. Как использовать Keras
3. Распределенные вычисления
4. **Практика: нейросеть на Keras**

MLP