**Министерство образования и науки Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение**
**высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ _____Информатика и системы управления_____

КАФЕДРА _____ Системы обработки информации и управления_____

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
## *К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ*

# Transfer Learning with tensorflow hub

Студент ИУ5-31М **Мьоу Зо У** _____
(Группа) (Подпись, дата) (Фамилия.И.О)

Руководитель **Юрий Евгеньевич Гапанюк**
(Подпись,дата) (фамилия.И.О)

**г. Москва, 2021 г**

**Министерство образования и науки Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение**
**высшего профессионального образования**
**«Московский государственный технический университет имени Н.Э. Баумана»**
**(МГТУ им. Н.Э.Баумана)**

УТВЕРЖДАЮ
Заведующий кафедрой ИУ-5
(Индекс)
В.М.Черненьки
(И.О.Фамилия)
« __ » ____ 2021г.

**З А Д А Н И Е**
**на выполнение научно-исследовательской работы**
**по теме«обработке и анализу данных»**

Студент группы _____ИУ5-31М_____
_____Мьоу Зо У_____
(фамилия, имя, отчество)
Направленость НИР (учебная, исследовательская, практическая производственная, и др.) исследовательская _____
Источник тематики (кафедра, предприятие, НИР) кафедра_____
График выполнения НИР: 25% к_нед, 50% к___нед, 75% к__нед, 100%к____нед.
***Техническое задани:*__ Transfer Learning with tensorflow hub**

***Офорление научно-исследовательской работы:***
Расчетно-пояснительная запимка на _____листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плпкаты, слайды и.т.д.)_____
Дата выдачи « ___» _____2021г.
Руководитель НИР ____ **Юрий Евгеньевич Гапанюк** _____
_____
*(Подпись,дата)* (*И.О.Фамилия)*
Студент _____ Мьоу Зо У
*(Подпись,дата)* (*И.О.Фамилия)*

# Contents

## 1. Введение

На самом деле модель Inception - извлечение полезной информации из изображения. Таким образом, мы можем вместо этого обучить начальную модель, используя другой набор данных. Но для того, чтобы полностью подготовить модель Inception к новому набору данных, требуется несколько недель, используя очень мощный и дорогой компьютер. Вместо этого мы можем повторно использовать предварительно обученную начальную модель и просто заменить слой, который выполняет окончательную классификацию. Это называется трансферным обучением(Transfer Learning).

## 2. Получение данных

Задача, которую мы собираемся решить в этой главе, - это проблема классификации набора данных цветов, которая доступна в наборах тензорных потоков (tfds). Имя набора данных istf_flowers, и оно состоит из изображений пяти разных видов цветов в разных разрешениях. Используя tfds, сбор данных является простым, и мы можем получить информацию о наборе данных, посмотрев на переменную info, возвращаемую вызовом tfds.load,

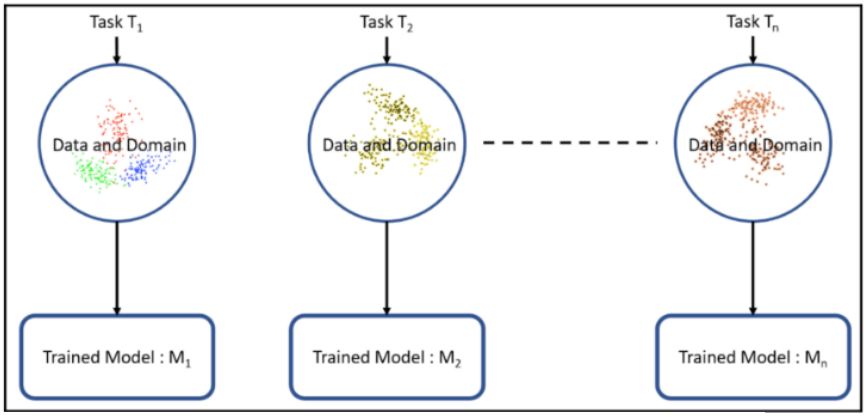
```
import tensorflow_datadet as tfds
dataset, info = tfds.load("tf_flower", with_info=True)
print(info)
```

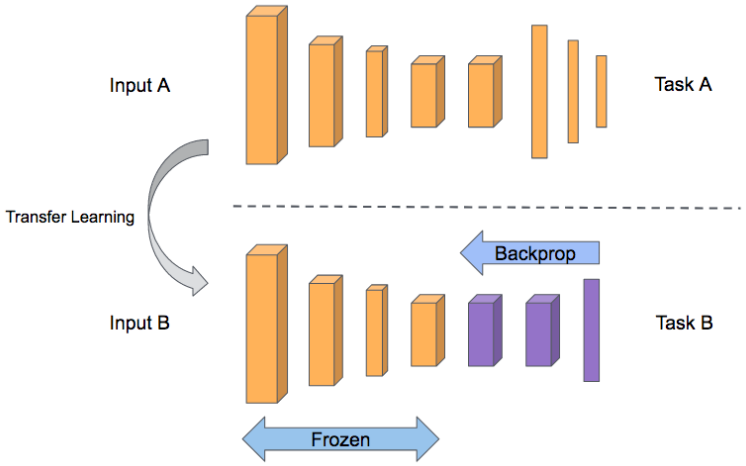
**Dataset**

```
data_root = tf.keras.utils.get_file(
'flower_photos','https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz',
  untar=True)
```


## 3. Введение в трансферное обучение

Традиционно алгоритмы обучения предназначены для самостоятельного решения задач или проблем. В зависимости от требований варианта использования и имеющихся данных применяется алгоритм для обучения модели для данной конкретной задачи. Традиционное машинное обучение (ML) обучает каждую модель изолированно на основе конкретной области, данных и задачи, как показано на следующем рисунке:




Трансферное обучение продвигает процесс обучения на один шаг вперед и в большей степени учитывает, как люди используют знания для выполнения задач. Таким образом, трансферное обучение - это метод повторного использования модели или знаний для другой связанной задачи. Трансферное обучение иногда также рассматривается как расширение существующих алгоритмов ML. Обширные исследования и работа проводятся в контексте трансферного обучения и понимания того, как знания могут передаваться между задачами. Тем не менее, семинар «Neural Information Processing Systems» (NIPS) 1995 года «Учиться учиться: консолидация и передача знаний в индуктивных системах», как полагают, обеспечил первоначальные мотивы для исследований в этой области.

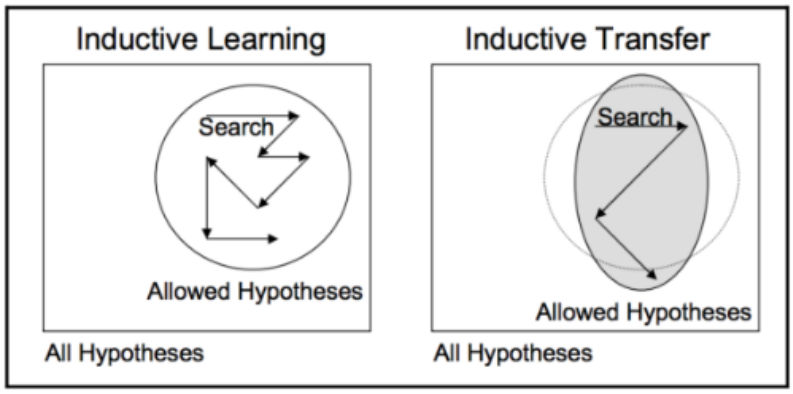
## 4. Трансферное обучение и глубокое обучение

Модели глубокого обучения представляют то, что также известно как индуктивное обучение. Цель алгоритмов индуктивного обучения - вывести отображение из набора обучающих примеров. Например, в случае классификации модель изучает сопоставление между входными объектами и

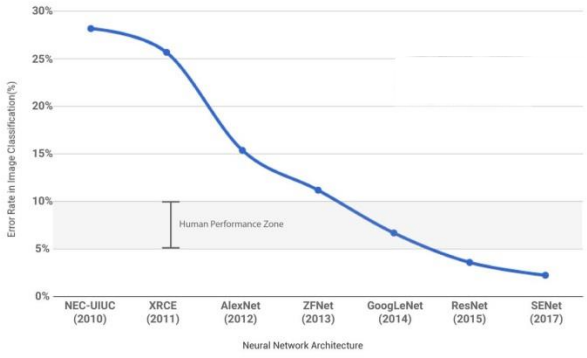
метками классов. Для того, чтобы такой учащийся хорошо обобщил невидимые данные, его алгоритм работает с набором предположений, связанных с распределением обучающих данных. Эти наборы предположений известны как индуктивное смещение.

Индуктивное смещение или допущения могут характеризоваться несколькими факторами, такими как пространство гипотез, которое он ограничивает, и процесс поиска в пространстве гипотез. Таким образом, эти предубеждения влияют на то, как и что изучается моделью для данной задачи и области.

Методы индуктивной передачи используют индуктивные смещения исходной задачи, чтобы помочь целевой задаче. Это может быть сделано разными способами, например, путем корректировки индуктивного смещения целевой задачи путем ограничения пространства модели, сужения пространства гипотез или внесения корректировок в сам процесс поиска с помощью знаний из исходной задачи. Этот процесс визуально изображен на следующей диаграмме:




**Transfer Learning**




Transfer learning- это заимствование архитектуры CNN с ее предварительно обученными параметрами у кого-то еще. Когда мы обучаем наши собственные данные на вершине предварительно обученных параметров.

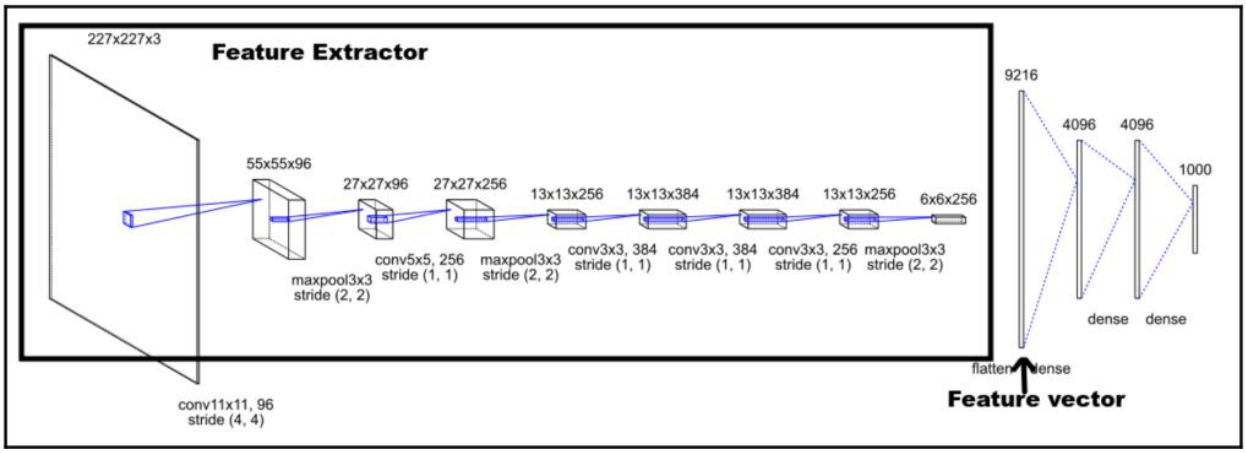
Только академические круги и некоторые отрасли промышленности обладают необходимым бюджетом и вычислительной мощностью для

обучения всего CNN с нуля, начиная со случайных весов, для массивного набора данных, такого как ImageNet. Поскольку эта дорогостоящая и трудоемкая работа уже проделана, целесообразно повторно использовать части обученной модели для решения нашей проблемы классификации. Фактически, можно перенести то, что сеть узнала из одного набора данных, в новый, передав таким образом знания. Трансферное обучение - это процесс обучения новой задаче, опираясь на ранее изученную задачу: процесс обучения может быть более быстрым, более точным и требовать меньшего количества обучающих данных. Идея трансферного обучения является яркой, и ее можно успешно применять при использовании сверточных нейронных сетей. Фактически, все сверточные архитектуры для классификации имеют фиксированную структуру, и мы можем повторно использовать их части в качестве строительных блоков для наших приложений. Общая структура состоит из трех элементов:

Входной слой Архитектура предназначена для приема изображения с точным разрешением. Входное разрешение влияет на всю архитектуру. Если разрешение входного слоя высокое, сеть будет глубже.

Экстрактор объектов: это набор свертки, объединения, нормализации и любого другого слоя, который находится между входным слоем и первым плотным слоем. Архитектура учится суммировать всю информацию, содержащуюся во входном изображении, в низкоразмерном представлении (на следующей диаграмме изображение размером 227 x 227 x 3 проецируется в 9216-мерный вектор).

Слои классификации: это стек полностью связанных слоев - полностью связанный классификатор, построенный поверх низкоразмерного представления входных данных, извлеченных классификатором:




Перенос знаний обученной модели в новую требует от нас удаления определенной части сети (которая является классификационными слоями) и сохранения CNN в качестве экстрактора признаков. Этот подход позволяет

нам использовать средство извлечения предварительно обученной модели в качестве строительного блока для нашей новой архитектуры классификации. При выполнении трансферного обучения предварительно обученная модель сохраняется постоянной, в то время как только новые слои классификации, прикрепленные поверх вектора признаков, являются обучаемыми. Таким образом, мы можем обучить классификатор, повторно используя знания, полученные в массиве массивных данных, и встраивая их в модель. Это приводит к двум значительным преимуществам:

- Это ускоряет процесс обучения, так как количество обучаемых параметров мало
- Это потенциально смягчает проблему переоснащения, поскольку извлеченные функции происходят из другого домена, и процесс обучения не может заставить их изменить

Все идет нормально. Идея трансферного обучения является яркой, и она может помочь решить несколько реальных проблем, когда наборы данных невелики и ресурсы ограничены. Единственная недостающая часть, которая также оказывается самой важной, это: где мы можем найти предварительно обученные модели? По этой причине команда TensorFlow создала TensorFlow Hub.

## 5. TensorFlow Hub

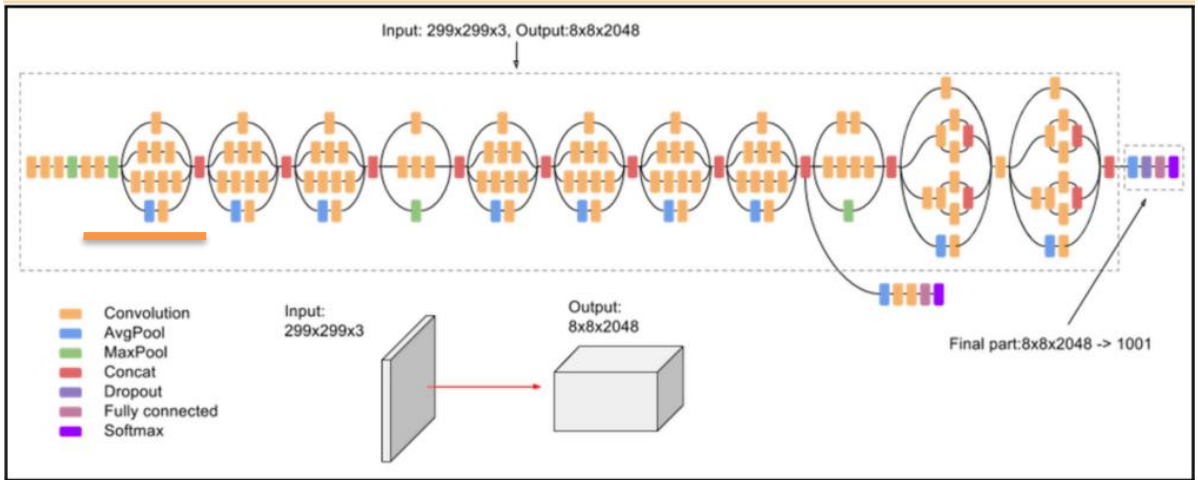
Описание TensorFlow Hub, которое можно найти в официальной документации, описывает, что такое TensorFlow Hub и о чем он довольно неплохо: TensorFlow Hub - это библиотека для публикации, обнаружения и использования повторно используемых частей моделей машинного обучения. Модуль - это отдельная часть графа TensorFlow вместе с его весами и активами, которые можно повторно использовать в различных задачах в процессе, известном как трансферное обучение. Передача обучения может:

- Обучение модели с меньшим набором данных
- улучшить обобщение и
- ускорить обучение

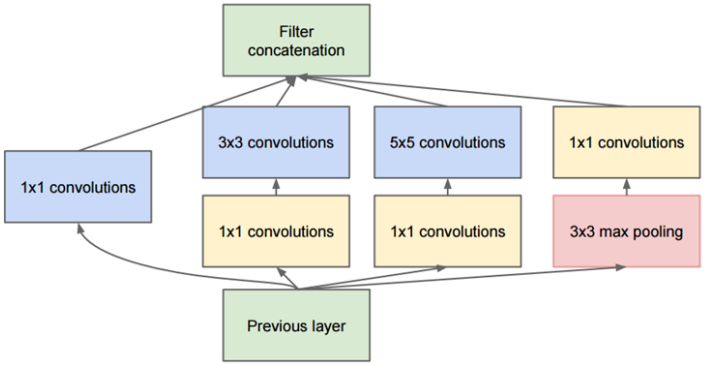
## 6. Использование Inception v3 в качестве средства извлечения функций

Полный анализ архитектуры Inception v3 выходит за рамки этой книги; однако, стоит отметить некоторые особенности этой архитектуры, чтобы правильно использовать ее для передачи обучения в другом наборе данных.

Inception v3 - это глубокая архитектура с 42 уровнями, победившая в конкурсе крупномасштабных визуальных изображений ImageNet (ILSVRC).



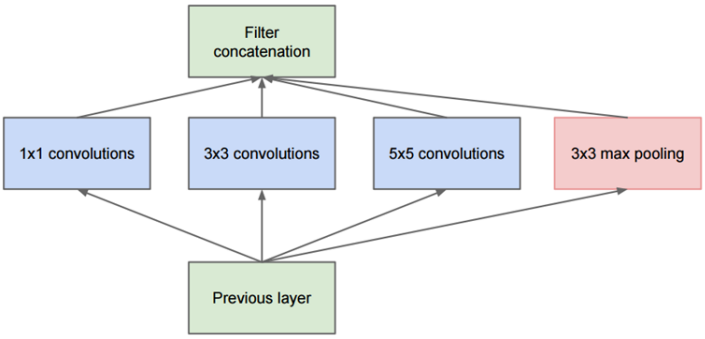

Это поле называется начальным модулем. Давайте внимательнее посмотрим, из чего он сделан.



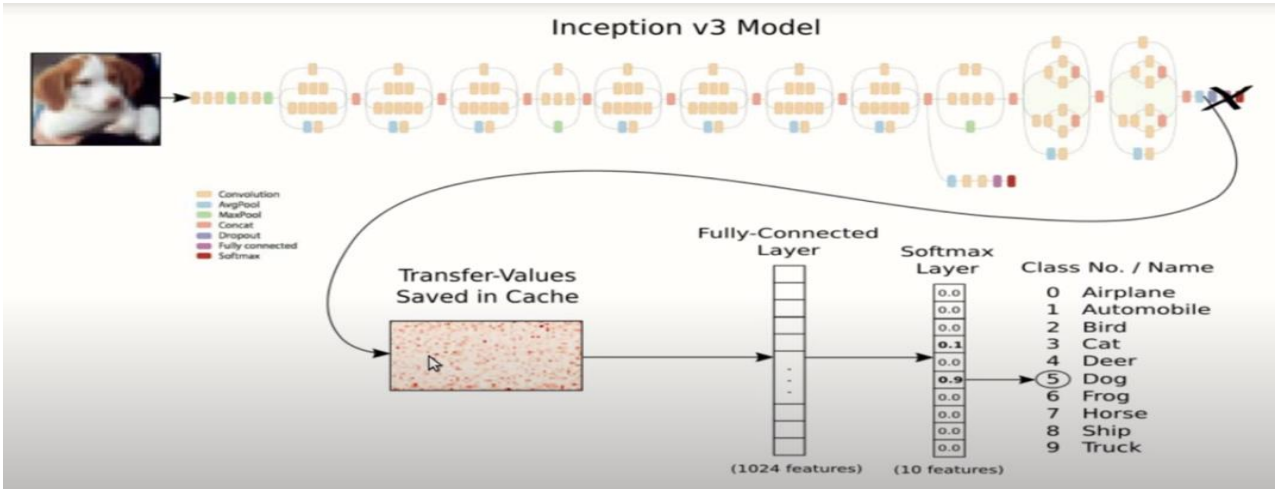

Full Inception module

Подчеркивание - это наши входные данные, а верхнее - выход модели (поворот этого изображения вправо на 90 градусов позволит вам визуализировать модель по отношению к последнему рисунку, который показывает всю сеть). По сути, на каждом уровне традиционной ConvNet вы должны сделать выбор, использовать ли операцию объединения или операцию conv (есть также выбор размера фильтра). Модуль Inception позволяет вам выполнять все эти операции параллельно. На самом деле это

была именно та «наивная» идея, которую придумали авторы.




Naïve idea of an Inception module

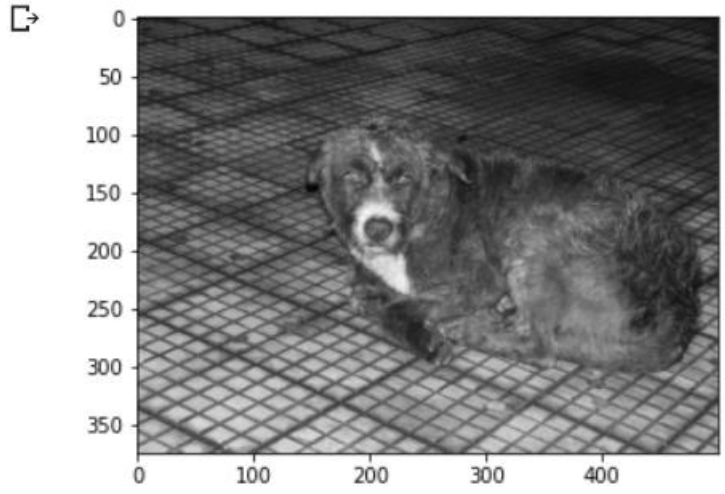



Сеть ожидает входное изображение с разрешением 299 x 299 x 3 и создает карту объектов 8 x 8 x 2048. Он был обучен на 1000 классов набора данных ImageNet, и входные изображения были масштабированы в диапазоне [0,1].

## 7. Трансферное обучение

Трансферное обучение - это процесс изучения новой задачи, основанный на ранее изученной задаче: процесс обучения может быть быстрее и точнее и требует меньше данных для обучения.

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2

DATADIR='/content/drive/MyDrive/img_data'
CATEGORIES=['dog','dog']
for category in CATEGORIES:
  path=os.path.join(DATADIR,category)

  for category in CATEGORIES:
    path = os.path.join(DATADIR,category) # path to nerd or beetle
    for img in os.listdir(path):
      img_array = cv2.imread(os.path.join(path,img) , 0)
      plt.imshow(img_array, cmap="gray")
      plt.show()

      break
    break
```



```
print(img_array)
```

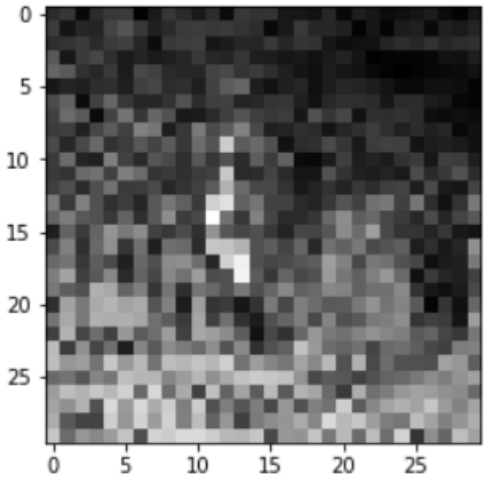
```
[[ 50  62  62 ...  33  32  33]
 [ 39  36  41 ...  36  38  40]
 [ 32  23  15 ...  31  27  26]
 ...
 [191 184 191 ... 165 166 172]
 [183 193 192 ... 171 163 171]
 [147 174 192 ... 163 166 166]]
```

```
print(img_array.shape)
```

```
(375, 500)
```

```
img_size=30
new_array=cv2.resize(img_array,(img_size,img_size))
plt.imshow(new_array,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f90e70c5650>
```



## 7.1 Простое трансферное обучение

Data Set

```
data_root = tf.keras.utils.get_file(
  'flower_photos','https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz',
   untar=True)
```

```
[40] image_generator = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)
     image_data = image_generator.flow_from_directory(str(data_root), target_size=IMAGE_SHAPE)
```

```
Found 3670 images belonging to 5 classes.
```

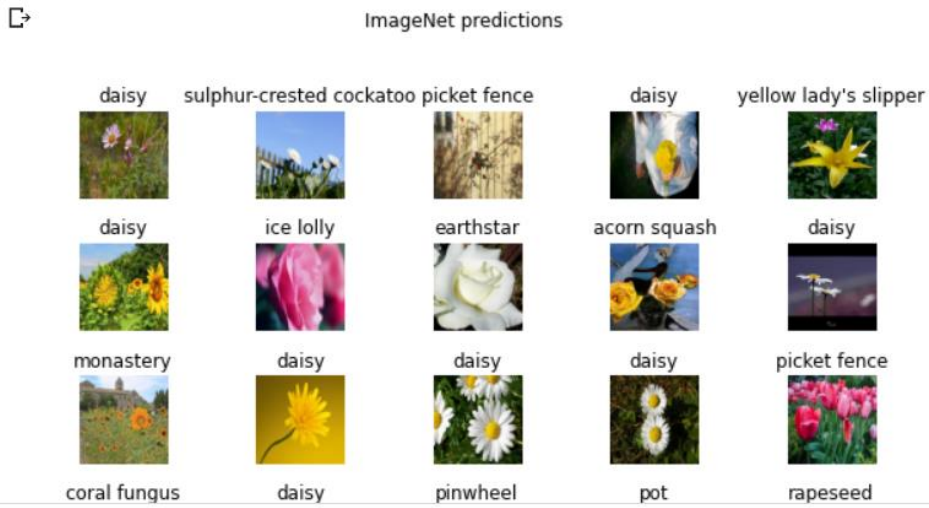
```
[41] for image_batch, label_batch in image_data:
       print("Image batch shape: ", image_batch.shape)
       print("Label batch shape: ", label_batch.shape)
       break
```

```
Image batch shape:  (32, 224, 224, 3)
Label batch shape:  (32, 5)
```

## 7.2 Запустите классификатор на пакете изображений

Что еще более важно, необходимо знать, на каком наборе данных была обучена сеть, трансферное обучение работает хорошо, если исходный наборданных имеет общие характеристики с целевым (новым) набором данных.

```
result_batch = classifier.predict(image_batch)
predicted_class_names = imagenet_labels[np.argmax(result_batch, axis=-1)]
plt.figure(figsize=(10,9))
plt.subplots_adjust(hspace=0.5)
for n in range(30):
  plt.subplot(6,5,n+1)
  plt.imshow(image_batch[n])
  plt.title(predicted_class_names[n])
  plt.axis('off')
_ = plt.suptitle("ImageNet predictions")
```



## 7.3 Присоедините классификационную головку

```
feature_extractor_url = "https://hub.tensorflow.google.cn/google/tf2
feature_extractor_layer = hub.KerasLayer(feature_extractor_url,
                                         input_shape=(224,224,3))
feature_batch = feature_extractor_layer(image_batch)
print(feature_batch.shape)  # (32, 1280)
feature_extractor_layer.trainable = False
```

```
(32, 1280)
```

```
[50] model = tf.keras.Sequential([
       feature_extractor_layer,
       layers.Dense(image_data.num_classes, activation='softmax')
     ])
```

```
[51] predictions = model(image_batch)
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_1 (KerasLayer)  (None, 1280)              2257984

 dense (Dense)               (None, 5)                 6405

=================================================================
```

```
predictions.shape
```

```
TensorShape([32, 5])
```

# 7.4 Обучение модели

```
model.compile(
  optimizer=tf.keras.optimizers.Adam(),
  loss='categorical_crossentropy',
  metrics=['acc'])
```

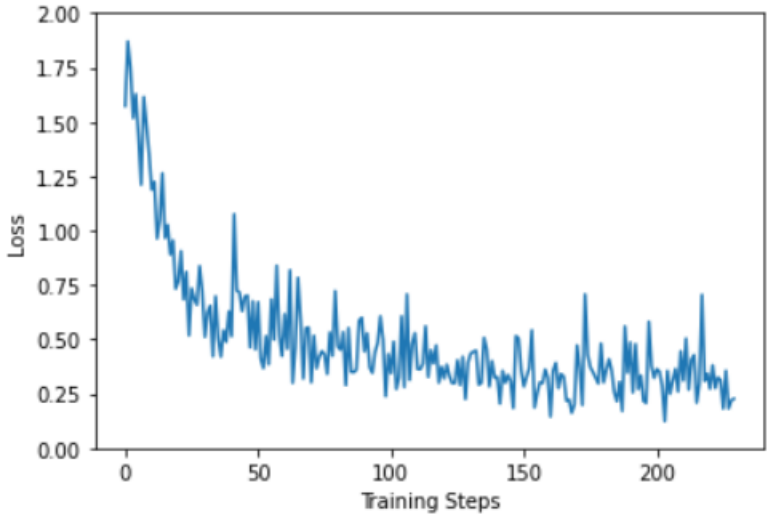
```
[55] class CollectBatchStats(tf.keras.callbacks.Callback):
       def __init__(self):
         self.batch_losses = []
         self.batch_acc = []

       def on_train_batch_end(self, batch, logs=None):
         self.batch_losses.append(logs['loss'])
         self.batch_acc.append(logs['acc'])
         self.model.reset_metrics()
```

```
steps_per_epoch = np.ceil(image_data.samples/image_data.batch_size)

batch_stats_callback = CollectBatchStats()

history = model.fit_generator(image_data, epochs=2,
                              steps_per_epoch=steps_per_epoch,
                              callbacks = [batch_stats_callback])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future
  import sys
Epoch 1/2
115/115 [==============================] - 105s 884ms/step - loss: 0.3270 - acc: 0.9062
Epoch 2/2
115/115 [==============================] - 101s 878ms/step - loss: 0.2273 - acc: 0.9375
```

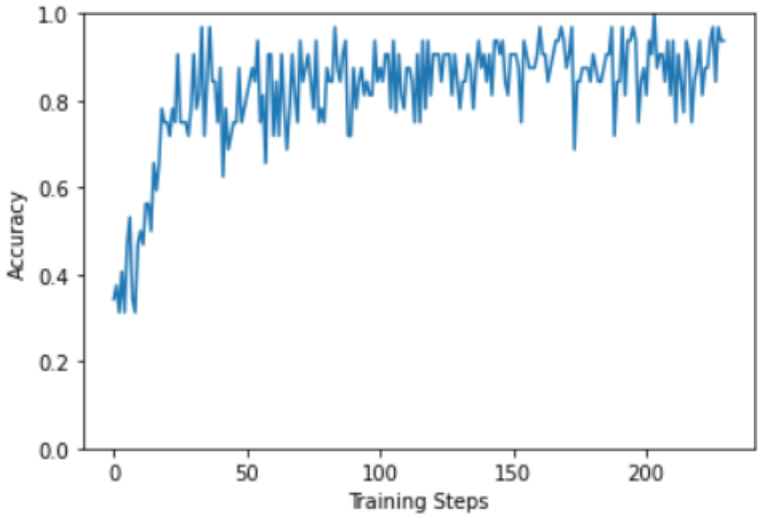
```
plt.figure()
plt.ylabel("Loss")
plt.xlabel("Training Steps")
plt.ylim([0,2])
plt.plot(batch_stats_callback.batch_losses)
```

```
[<matplotlib.lines.Line2D at 0x7f01c05609d0>]
```



```
# Change in precision
plt.figure()
plt.ylabel("Accuracy")
plt.xlabel("Training Steps")
plt.ylim([0,1])
plt.plot(batch_stats_callback.batch_acc)
```

```
[<matplotlib.lines.Line2D at 0x7f01c0648d90>]
```




## 7.5 Проверка прогноза

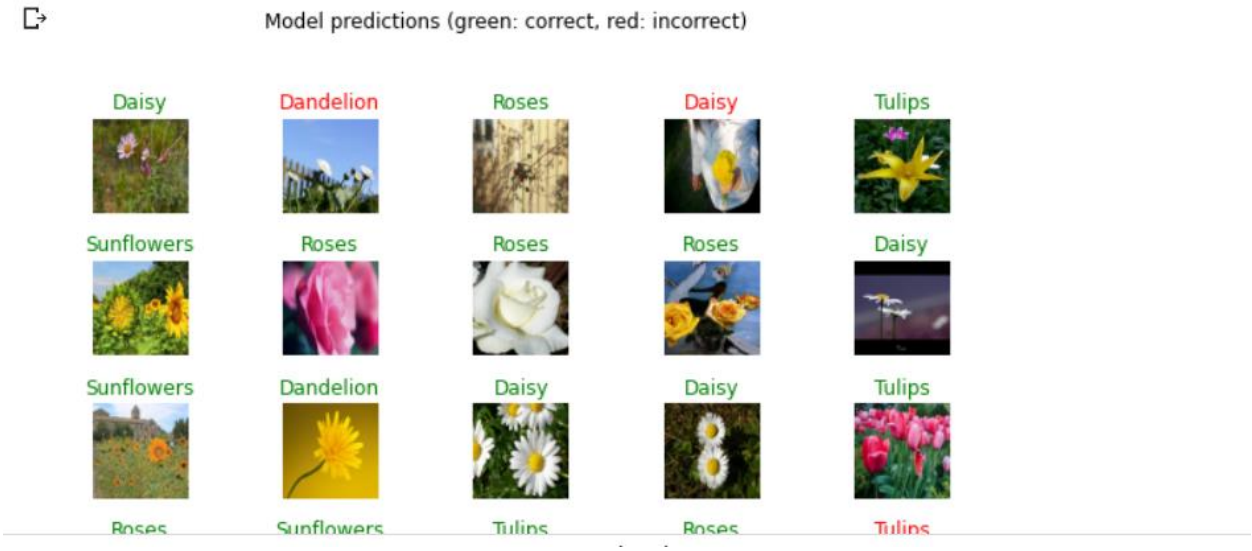
```
[64] # Test results
class_names = sorted(image_data.class_indices.items(), key=lambda pair:pair[1])
class_names = np.array([key.title() for key, value in class_names])
class_names
```

```
array(['Daisy', 'Dandelion', 'Roses', 'Sunflowers', 'Tulips'],
      dtype='<U10')
```

```
[65] predicted_batch = model.predict(image_batch)
predicted_id = np.argmax(predicted_batch, axis=-1)
predicted_label_batch = class_names[predicted_id]
label_id = np.argmax(label_batch, axis=-1)
```

```
label_id=np.argmax(label_batch,axis=-1)
```

```
# Visualization
plt.figure(figsize=(10,9))
plt.subplots_adjust(hspace=0.5)
for n in range(30):
  plt.subplot(6,5,n+1)
  plt.imshow(image_batch[n])
  color = "green" if predicted_id[n] == label_id[n] else "red"
  plt.title(predicted_label_batch[n].title(), color=color)
  plt.axis('off')
_ = plt.suptitle("Model predictions (green: correct, red: incorrect)")
```



Export your model

```
[68] import time
     t = time.time()

     export_path = "/tmp/saved_models/{}".format(int(t))
     model.save(export_path, save_format='tf')

     export_path
```

```
INFO:tensorflow:Assets written to: /tmp/saved_models/1636810854/assets
INFO:tensorflow:Assets written to: /tmp/saved_models/1636810854/assets
'/tmp/saved_models/1636810854'
```

```
[69] # Download
     reloaded = tf.keras.models.load_model(export_path)
```

```
[70] result_batch = model.predict(image_batch)
     reloaded_result_batch = reloaded.predict(image_batch)
```

```
[71] abs(reloaded_result_batch - result_batch).max()
```

```
0.0
```

## 8. Вывод

Используя набор данных Tensorflow, у нас есть вся информация, необходимая для простого выполнения как классификации, так и локализации, поскольку каждая строка представляет собой словарь, который содержит метку каждого процента ограничивающей рамки в изображении. Более того, поскольку набор данных отфильтрован, чтобы иметь только изображения с одним объектом внутри, мы можем пройти обучение классификационной головки точно так же, как обучение модели классификации.

# 9. Список литературы


1. https://books.google.ru/books?id=w5iwDwAAQBAJ&pg=PA197&lpg=PA197&dq=transfer+learning+to+train+your+network+on+your+data.+Model+you+can+get+from+tensorflow+hub&source=bl&ots=LBLGTLxFRY&sig=ACfU3U0b_Quu5zyVPyl6m1QojdZChPfPtQ&hl=en&sa=X&ved=2ahUKEwibh4yv-OfpAhVn2aYKHdLOD3kQ6AEwC3oECAUQAQ#v=onepage&q=transfer%20learning%20to%20train%20your%20network%20on%20your%20data.%20Model%20you%20can%20get%20from%20tensorflow%20hub&f=false
2. https://www.youtube.com/watch?v=j-3vuBynnOE
3. https://www.google.ru/search?tbm=bks&hl=en&q=transfer+learning+with+tensorflow+hub
4. https://adeshpande3.github.io/adeshpande3.github.io/