Experts

Gotchas of transfer learning for image classification

Sayak Paul
PyImageSearch
@RisingSayak

Google Developers

Ideal audience

- ML Developers that have trained an image classifier

Experts

Agenda

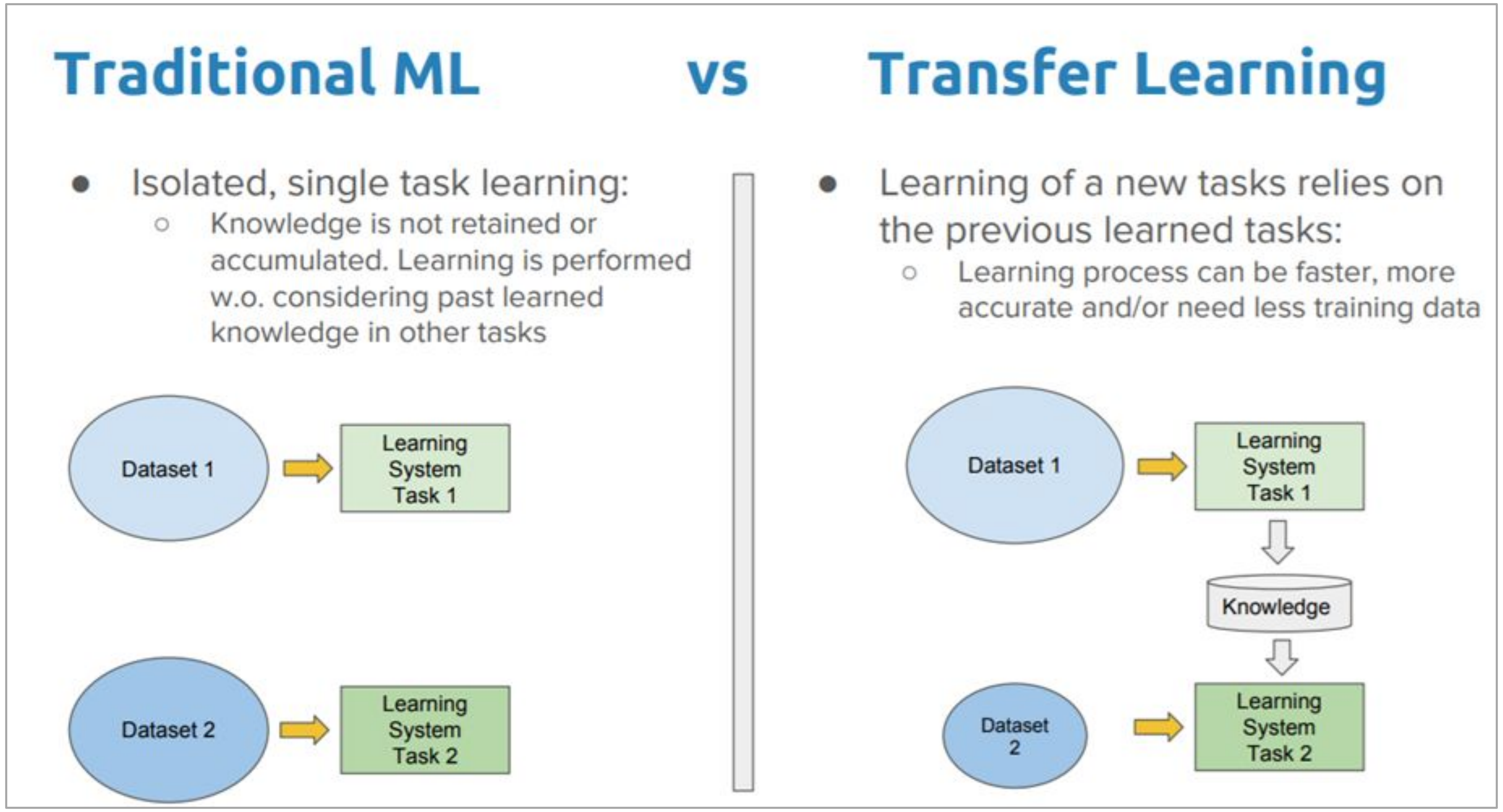
- Brief introduction to transfer learning
- The ImageNet moment
- Transfer learning for image classification
- Things to keep in mind while doing transfer learning
- QA

Transfer learning: Intro

Transfer learning: Intro

“After supervised learning — Transfer Learning will be the next driver of ML commercial success.” - Andrew NG (*source*)

Transfer learning: Intro

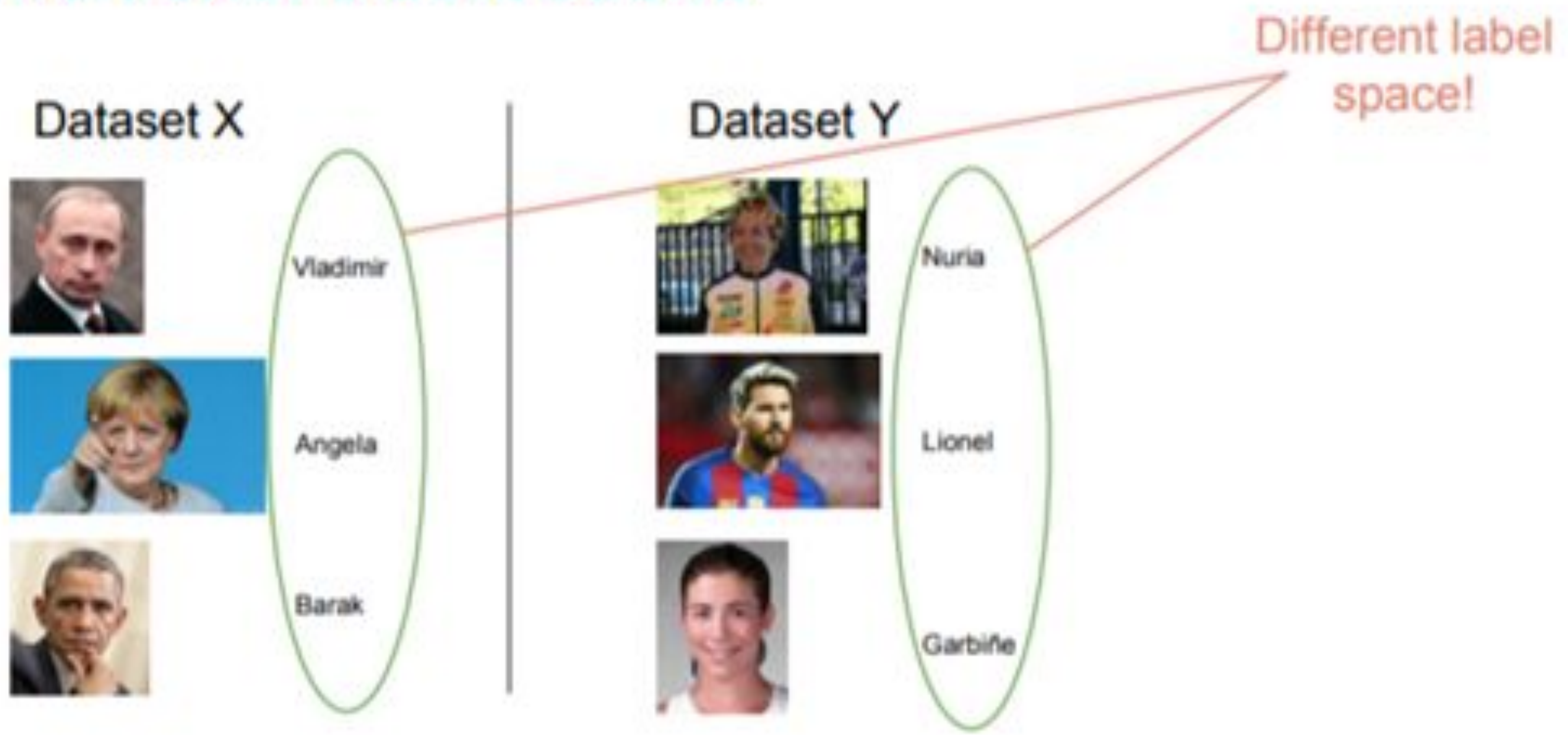


[Source]

Transfer learning: Intro

If two domains are different, they may have different feature spaces or different marginal distributions

If two tasks are different, they may have different label spaces or different conditional distributions



Source

Transfer learning: Intro

For a more formal treatment of transfer learning, refer to [Transfer Learning - Machine Learning's Next Frontier] by Sebastian Ruder.

The ImageNet moment

ImageNet Challenge

- 1,000 object classes (categories).
- Images:
    - 1.2 M train
    - 100k test.

[Source](#)

A seminal moment

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Source

A seminal moment

- It opened numerous new possibilities which previously seemed impossible.

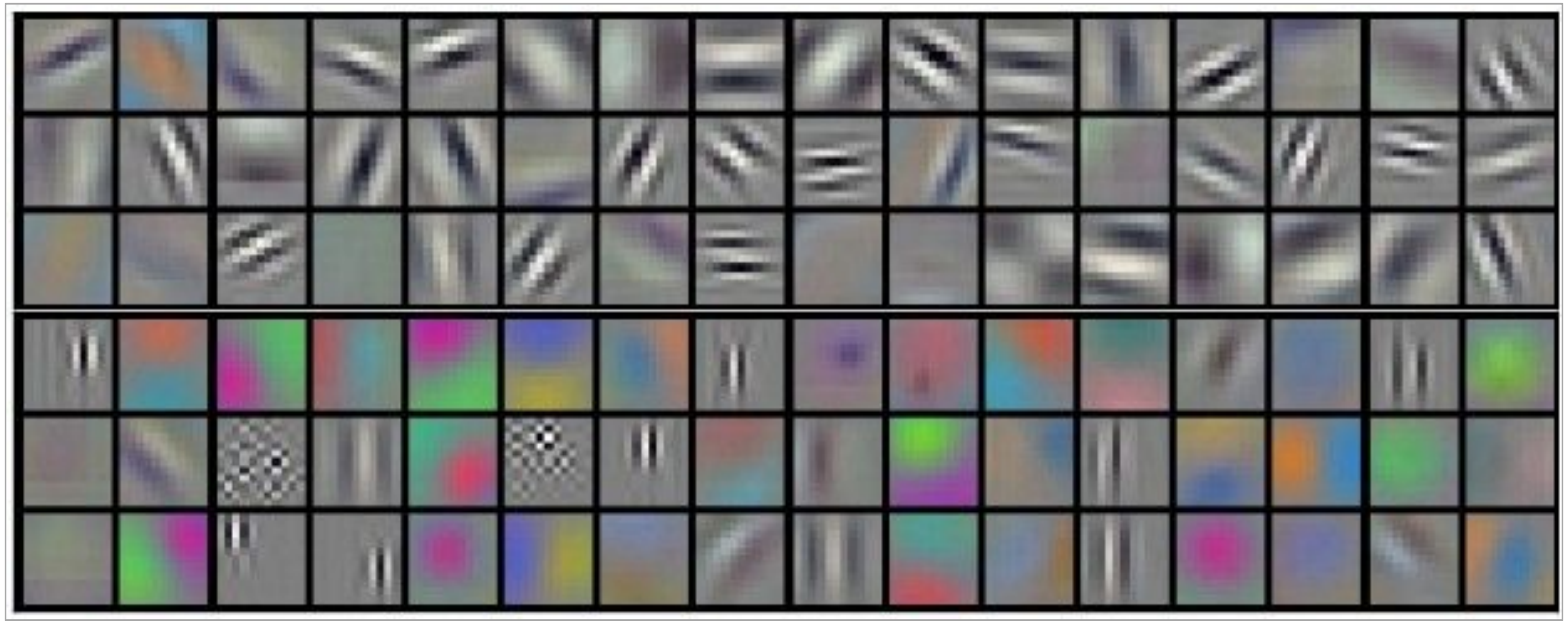
A seminal moment

- It opened numerous new possibilities which previously seemed impossible.
- Researchers developed models that achieved superhuman performance.

A seminal moment

- It opened numerous new possibilities which previously seemed impossible.
- Researchers developed models that achieved superhuman performance.
- The knowledge learned in those models turned out to be ***gold*** for computer vision tasks.

Representation learning ftw!



Source

Representation learning ftw!

- Turns out that these low-level features actually constitutes the basic structure of images.

Representation learning ftw!

- Turns out that these low-level features actually constitutes the basic structure of images.
- These low-level features help to develop an understanding of the underlying structures of the images.

Representation learning ftw!

- Turns out that these low-level features actually constitutes the basic structure of images.
- These low-level features help to develop an understanding of the underlying structures of the images.
- These representations are applicable to tasks having visual input modalities -

Representation learning ftw!

- Turns out that these low-level features actually constitutes the basic structure of images.
- These low-level features help to develop an understanding of the underlying structures of the images.
- These representations are applicable to tasks having visual input modalities -
  - Image classification

Representation learning ftw!

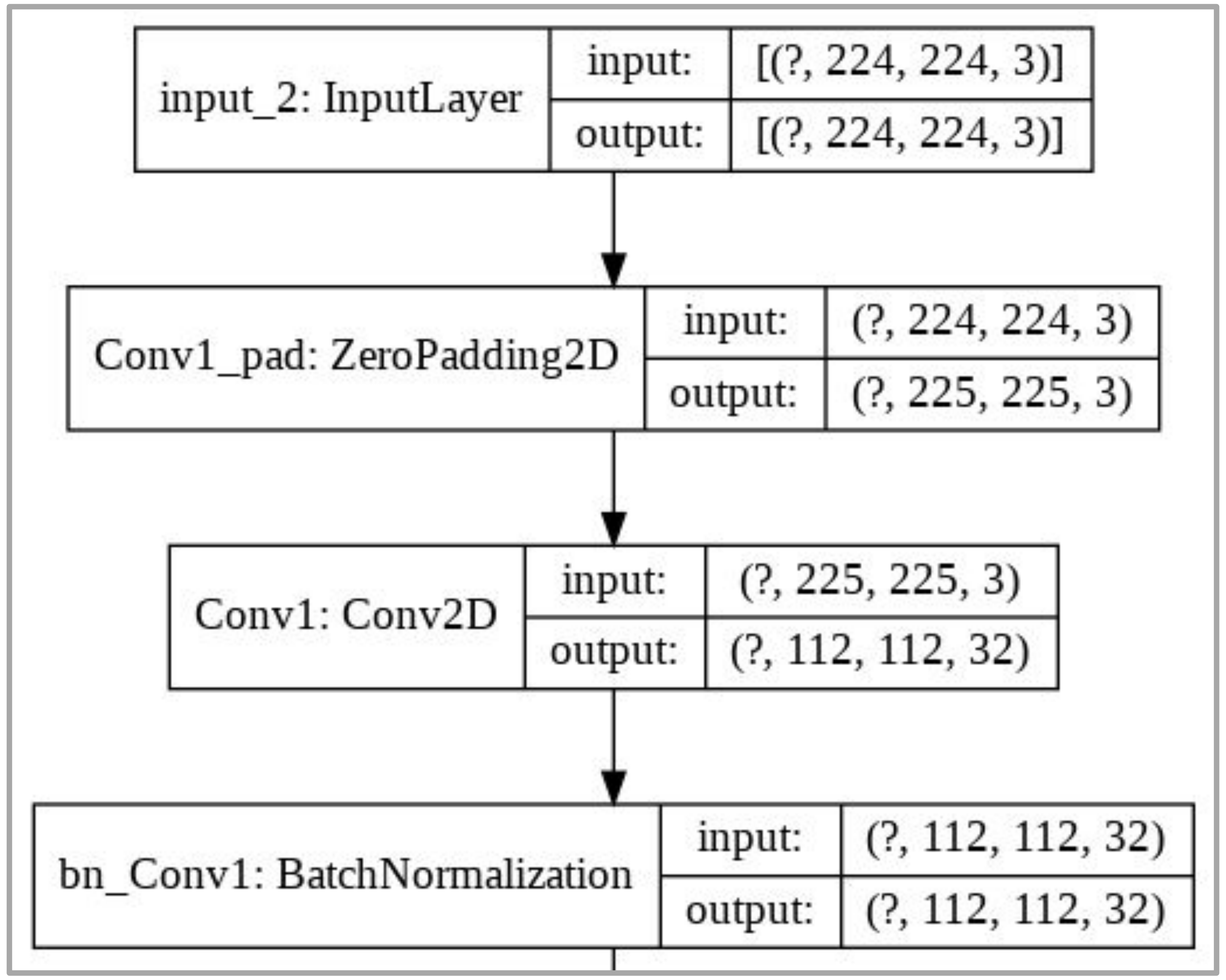
- ...
- These low-level features help to develop an understanding of the underlying structures of the images.
- These representations are applicable to tasks having visual input modalities -
  - Image classification
  - Object detection

Representation learning ftw!

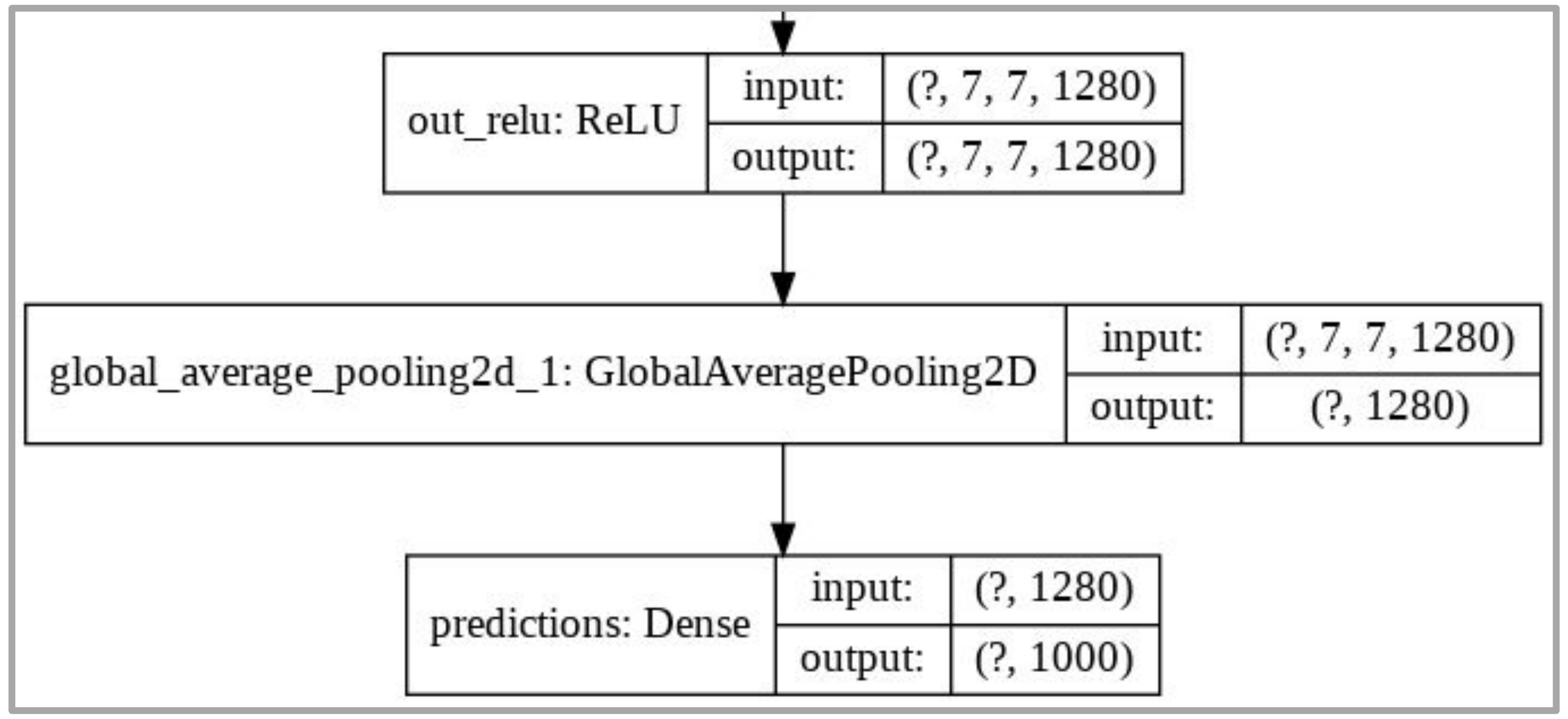
- ...
- These low-level features help to develop an understanding of the underlying structures of the images.
- These representations are applicable to tasks having visual input modalities -
  - Image classification
  - Object detection
  - Image captioning
  - and *many more*!

Transfer learning for image classification

A sample architecture (MobileNetV2) -



First half of the network



Classification top of the network

Transfer learning for image classification

Three recipes -

- Off-the-shelf inference.
- Precomputing bottlenecks and replacing the classification top.
- Fine-tuning.

Off-the-shelf inference

Experts

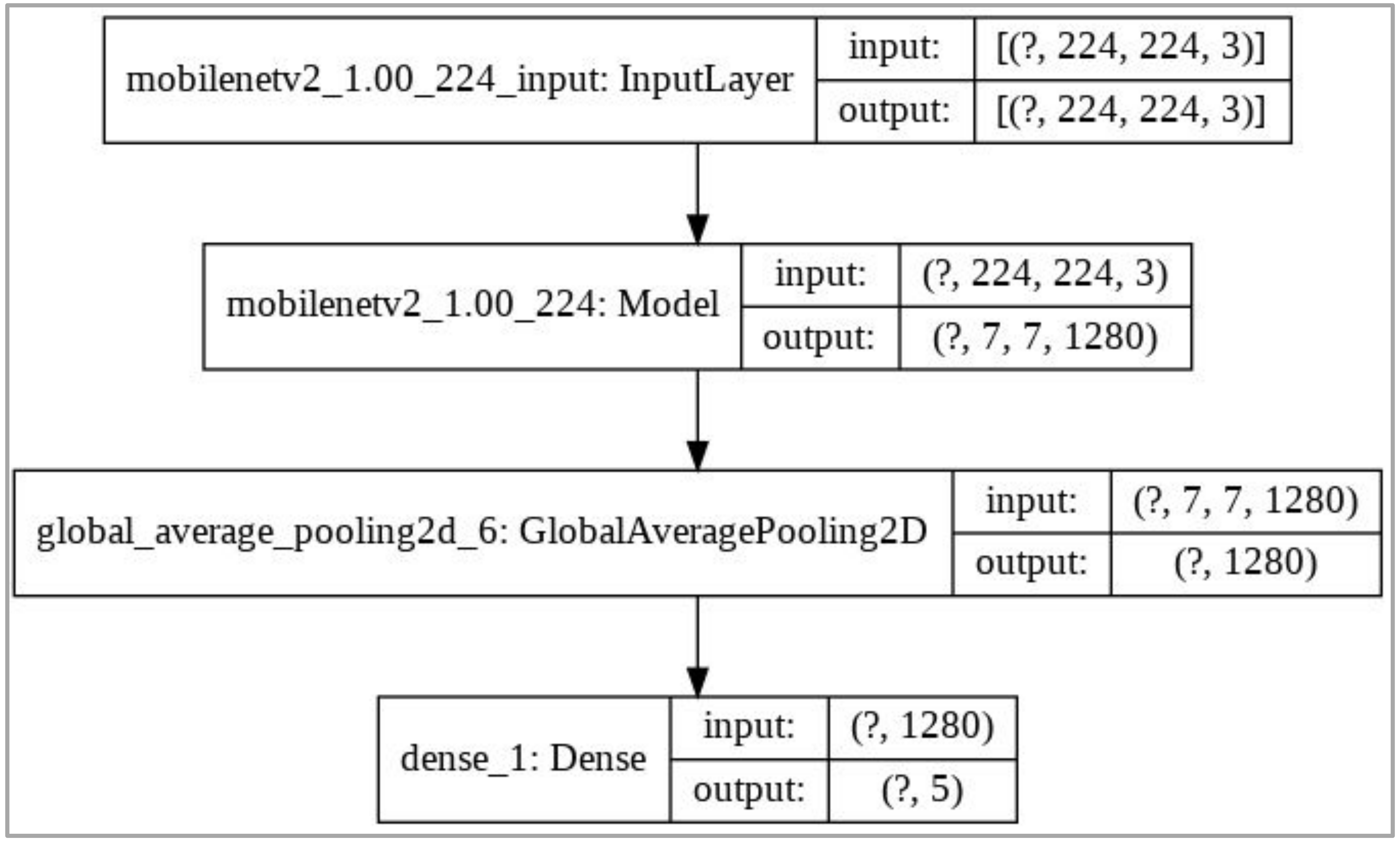
```python
# Load off the base model
mobilenet = MobileNetV2(weights="imagenet")

# Run inference and parse the predictions
result = mobilenet.predict(image)
print(decode_predictions(result, top=3)[0])
```

Precomputing ***bottlenecks*** and replacing the classification top

Experts

```python
# Load the pre-trained model without classification top
mobilenet = MobileNetV2(weights="imagenet", include_top=False)
# Set the base model to non-trainable
mobilenet.trainable = False
# Build the new model
new_model = Sequential([
    mobilenet,
    GlobalAveragePooling2D(),
    Dense(len(CLASSES), activation="softmax")
])
```

Resultant model



Resultant model

```
1 new_model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Model) (None, 7, 7, 1280)        2257984
_________________________________________________________________
global_average_pooling2d_6 ( (None, 1280)              0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 6405
=================================================================
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
_________________________________________________________________
```

Number of trainable parameters very low as expected

Fine-tuning

```
# Load the pre-trained model without classification top
mobilenet = MobileNetV2(weights="imagenet", include_top=False)
# Set the base model to *trainable*
mobilenet.trainable = True
# Build the new model
new_model_trainable = Sequential([
    mobilenet,
    GlobalAveragePooling2D(),
    Dense(len(CLASSES), activation="softmax")
])
```

Resultant model

```
1 new_model_trainable.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Model) (None, 7, 7, 1280)        2257984
_________________________________________________________________
global_average_pooling2d_7 ( (None, 1280)              0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 6405
=================================================================
Total params: 2,264,389
Trainable params: 2,230,277
Non-trainable params: 34,112
_________________________________________________________________
```

Number of trainable parameters
changes as expected

Fine-tuning

Can be done in two variants:

- After doing a round plain transfer learning.
- Training the pre-trained network from the beginning with **trainable=True**.

Things to keep in mind

- When pre-computing the bottlenecks/embeddings:
  - Be careful about allocating resources.

Things to keep in mind

- When pre-computing the bottlenecks/embeddings:
  - Be careful about allocating resources.

```
1 new_model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Model) (None, 7, 7, 1280)        2257984
_________________________________________________________________
global_average_pooling2d_6 ( (None, 1280)              0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 6405
=================================================================
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
_________________________________________________________________
```

You are only training a ***shallow fully-connected*** network.

Things to keep in mind

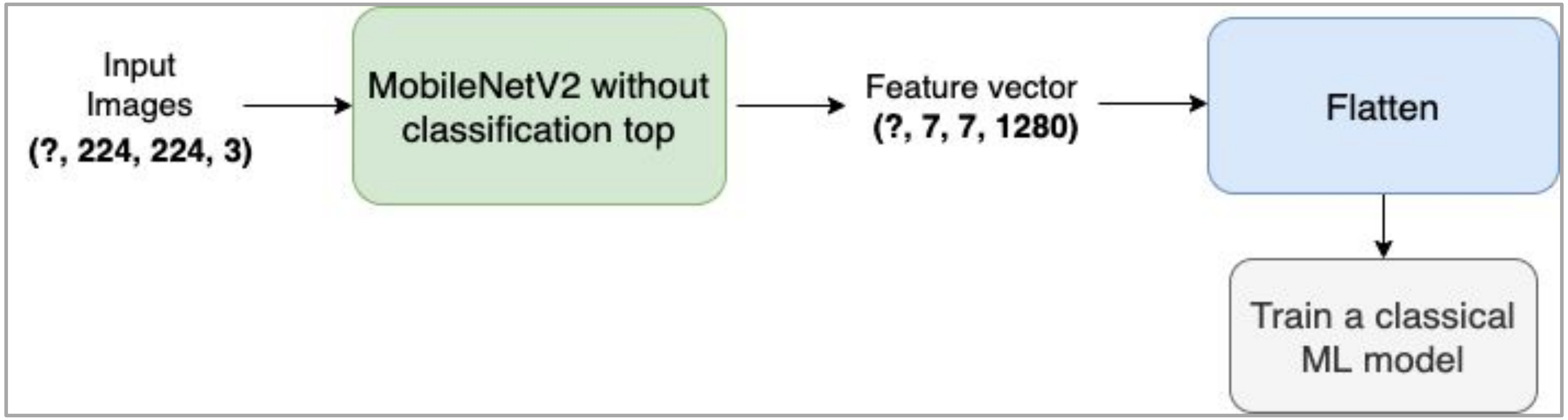
- When pre-computing the bottlenecks/embeddings:
  - Be careful about allocating resources.
  - Try with a ***relatively higher learning rate*** in this case.

Things to keep in mind

- Pre-computing the bottlenecks/embeddings.
- Pre-trained networks + Classical ML.

Things to keep in mind

- Pre-computing the bottlenecks/embeddings.
- Pre-trained networks + Classical ML.



An example [here]

Things to keep in mind

- Pre-computing the bottlenecks/embeddings.
- Pre-trained networks + Classical ML.
- When doing fine-tuning:
    - Start with a ***lower learning rate*** -

Things to keep in mind

- Pre-computing the bottlenecks/embeddings.
- Pre-trained networks + Classical ML.
- When doing fine-tuning:
  - Start with a ***lower learning rate*** -
    - Discriminative learning rates
    - Learning rate schedules

Things to keep in mind

- Pre-computing the bottlenecks/embeddings.
- Pre-trained networks + Classical ML.
- When doing fine-tuning?
- Mean subtraction with the source dataset mean stats.

```python
# Initialize your data generator
train_aug = ImageDataGenerator()

# ImageNet (source dataset) mean stats and assign
mean = np.array([123.68, 116.779, 103.939], dtype="float32")
train_aug.mean = mean
```

Things to keep in mind

- Pre-computing the bottlenecks/embeddings.
- Pre-trained networks + Classical ML.
- When doing fine-tuning?
- Mean subtraction with the source dataset mean stats.
- Set the batchnorm layers to be ***non-trainable***.

```python
# Freeze the base_model
base_model.trainable = False
# Create new model on top
inputs = keras.Input(shape=(150, 150, 3))
# Make sure the base model runs in *inference mode*
x = base_model(x, training=False)
# Classification top
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)
outputs = keras.layers.Dense(1)(x)
model = keras.Model(inputs, outputs)
```

Things to keep in mind

- Pre-computing the bottlenecks/embeddings.
- Pre-trained networks + Classical ML.
- When doing fine-tuning?
- Mean subtraction with the source dataset mean stats.
- Set the batchnorm layers to be ***non-trainable***.
- Gradual unfreezing of layers while doing fine-tuning.
  (Should be performed after a round of transfer learning)

```python
# Unfreeze some of the upper layers of base model
for layer in base_model.layers[15:]:
    layer.trainable = True


# Compile again and train
model.compile(...)
model.fit(...)
```

Very important!

New frontier: Self-supervised learning

- ImageNet is a labeled dataset.

New frontier: Self-supervised learning

- ImageNet is a labeled dataset.
- Affording such a huge labeled dataset is often not a feasible option for many industries (for example - medical).

New frontier: Self-supervised learning

- ImageNet is a labeled dataset.
- Affording such huge amount of labeled dataset is often not a feasible option for many industries (for example - medical).
- Unlabeled data, whereas, is way cheaper.

New frontier: Self-supervised learning

Check out ***SimCLR*** that presents a simple yet effective framework to utilize unlabeled data to learn useful representations from visual data.

Questions?

Experts

References

- A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning by Dipanjan Sarkar
- Transfer Learning - Machine Learning's Next Frontier by Sebastian Ruder
- Fine-tuning with Keras and Deep Learning by PyImageSearch
- Transfer learning notebook by François Chollet

Slides available here -

https://bit.ly/tl-sayak

Experts

Experts

Thank You!

Sayak Paul
PyImageSearch
@RisingSayak