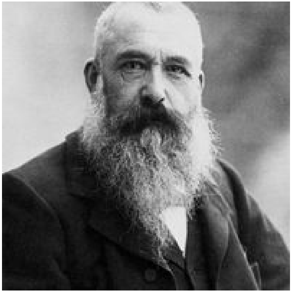
The Power of Transfer Learning in Artist Identification

Xingyu Zhou

The Ohio State University

April 30, 2018

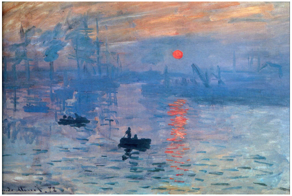
Warm Up



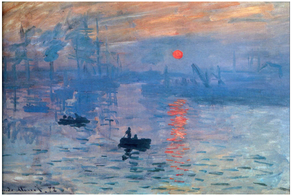
Claude Monet

Vincent Van Gogh

Try it yourself



Try it yourself



Claude Monet

Vincent Van Gogh

Try it yourself

Try it yourself

Claude Monet

Vincent Van Gogh

Can we solve it with machine learning?

Can we solve it with machine learning?

Let's try it

Data set

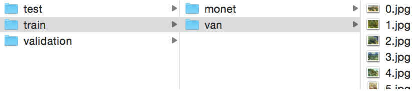
WIKIART

VISUAL ART ENCYCLOPEDIA

Data set

WIKIART
VISUAL ART ENCYCLOPEDIA

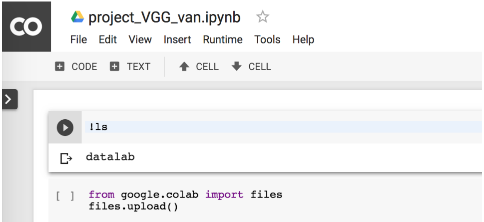
- We collect 300 images for *each* artist.
- We split into 240 for training, 30 for validation and 30 for testing.
- Folder structure:



Platform

Platform

- We use Keras with tensorflow backend to support neural networks.
- We use *Google Colaboratory* as our computing engine.
  - Free Tesla K80 GPU!
  - It is similar to Jupyter notebook:

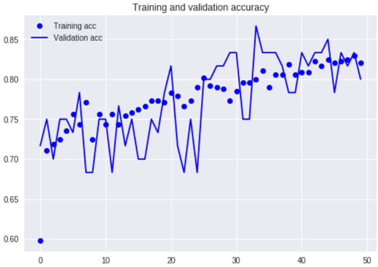


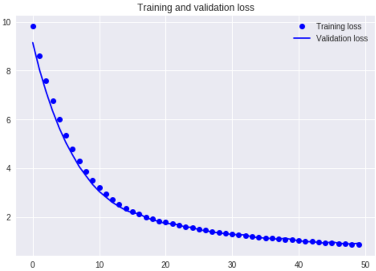
Baseline CNN

```
model.summary()
```

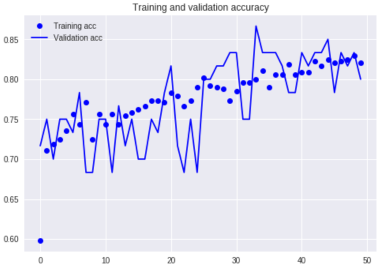
```
________________________________________________________________
Layer (type)                 Output Shape              Param #
================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 32)      896
________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 32)      0
________________________________________________________________
conv2d_2 (Conv2D)            (None, 109, 109, 64)      18496
________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 54, 54, 64)        0
________________________________________________________________
conv2d_3 (Conv2D)            (None, 52, 52, 128)       73856
________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 26, 26, 128)       0
________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 128)       147584
________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 12, 12, 128)       0
________________________________________________________________
flatten_1 (Flatten)          (None, 18432)             0
________________________________________________________________
dropout_1 (Dropout)          (None, 18432)             0
________________________________________________________________
dense_1 (Dense)              (None, 512)               9437696
________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
================================================================
Total params: 9,679,041
Trainable params: 9,679,041
Non-trainable params: 0
________________________________________________________________
```

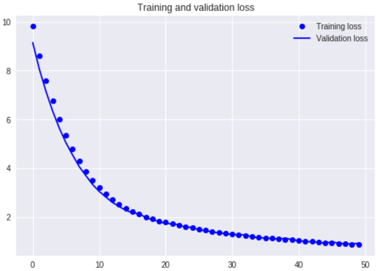
Baseline CNN





Baseline CNN

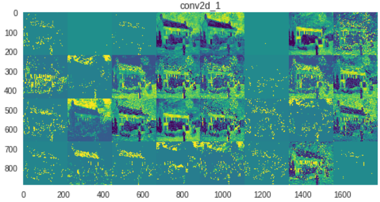




Test Accuracy: 83.3%

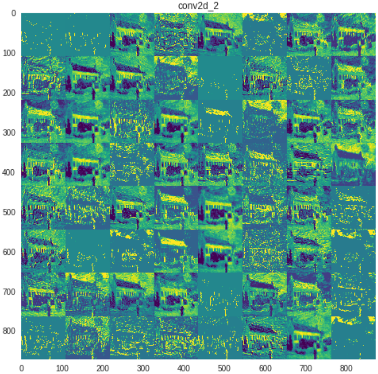
```
Found 60 images belonging to 2 classes.
test acc: 0.8333333233992258
```

Activations Visualization

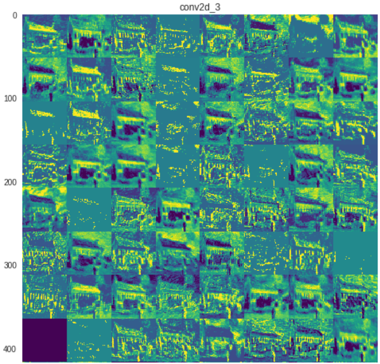


First layer: it keeps almost all of the information in the initial image.

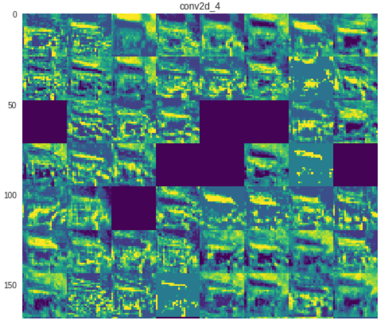
Activations Visualization



Activations Visualization



Activations Visualization



Higher layers:

- activations become *abstract*, less information about the visual contents.
- the sparsity of the activations increases.

Well, the result is okay, but definitely not perfect, is it?

Well, the result is okay, but definitely not perfect, is it?

I agree, let's improve it!

Transfer Learning

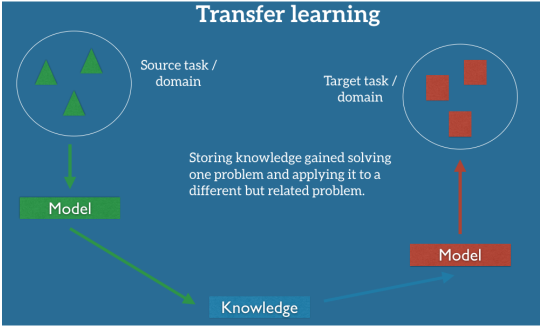


Figure: Transfer learning setup[1]

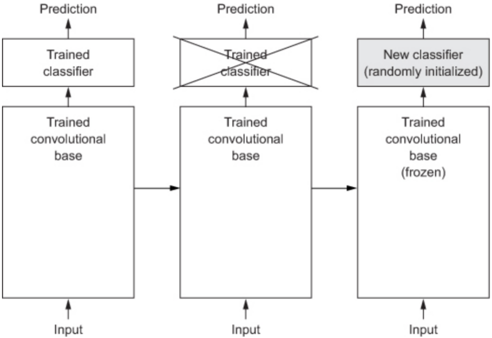
Source:

- Task: ImageNet
- Model: VGG16

Target:

- Task: Artist identification
- Model: Softmax classifier

Transfer learning



In our case:

- Trained convolutional base: VGG16
- New classifier: Softmax.

Transfer learning with VGG16

- Load pre-trained VGG16 model as base.

```
[ ]  from keras.applications import VGG16

     conv_base = VGG16(weights='imagenet',
                       include_top=False,
                       input_shape=(224, 224, 3))
```

- Extract features

```
[ ]  train_features = np.reshape(train_features, (480, 7 * 7 * 512))
     validation_features = np.reshape(validation_features, (60, 7 * 7 * 512))
     test_features = np.reshape(test_features, (60, 7 * 7 * 512))
```

```
[ ]  np.shape(train_features)
```

```
(480, 25088)
```

- Add densely connected classifier

```
from keras import models
from keras import layers
from keras import optimizers
from keras import regularizers

model = models.Sequential()
model.add(layers.Dense(256, kernel_regularizer=regularizers.l2(5e-5),
                       activation='relu', input_dim=7 * 7 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

Before we start...

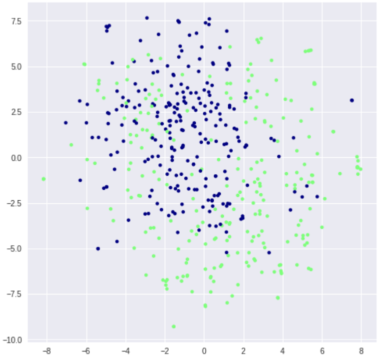
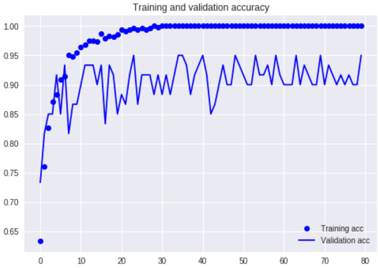
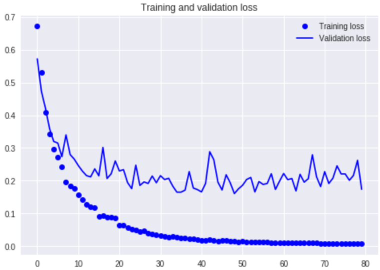


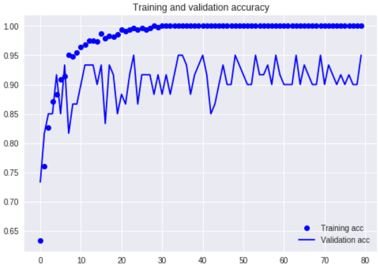
Figure: Bottleneck features visualization, created by *tsne*

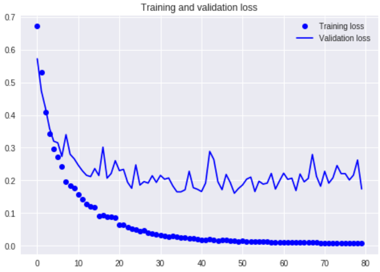
Now, let's train it...





Now, let's train it...





Test Accuracy: 94.6%

Nice result, but, shall we be satisfied with it?

Nice result, but, shall we be satisfied with it?

No, let's further improve it!

Fine tuning

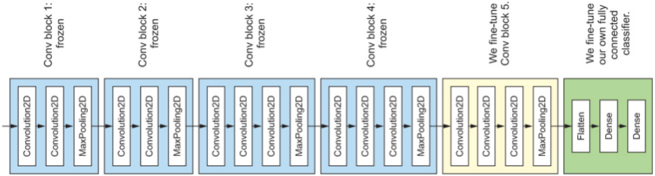
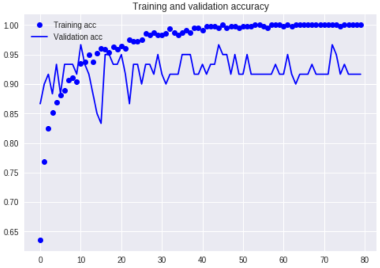
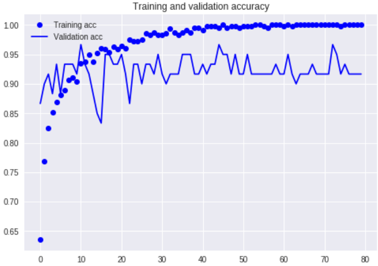


Figure: Frozen certain layers and fine tuning certain blocks only.

Show me the result...



Show me the result...

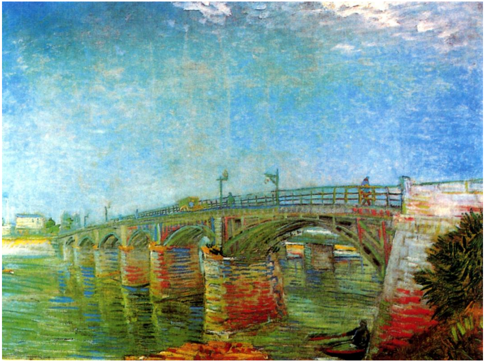


Test Accuracy: 98.3% (make mistake on just one image) 

```
test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(224, 224),
        batch_size=10,
        class_mode='binary')
test_loss, test_acc = model.evaluate_generator(test_generator,steps = 6)
test_acc
```

```
Found 60 images belonging to 2 classes.
0.9833333293596903
```

The missing one...



Does it really learn?

Some technicals

- Use data processing and augmentation.
- Batch size is 10.
- Training for 80 epochs.
- Optimizer is RMSprop with learning rate $2e^{-5}$.
- Activation is ReLu.
- L2 regularization of $1e^{-5}$.
- Dropout with probability 0.5.

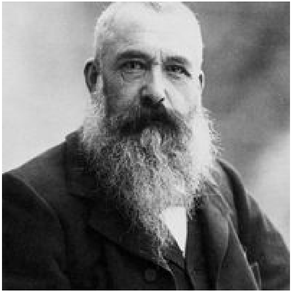
Really nice, but, wait a minute...can you distinguish which column is Monet's works?

Really nice, but, wait a minute...can you distinguish which column is Monet's works?

Claude Monet

Alfred Sisley

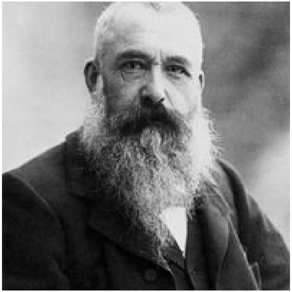
A harder problem...



Claude Monet

Alfred Sisley

A harder problem...



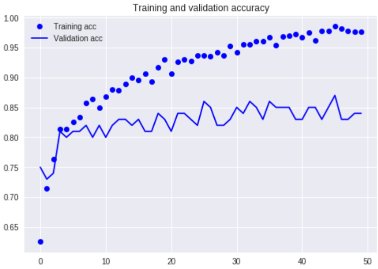
Claude Monet

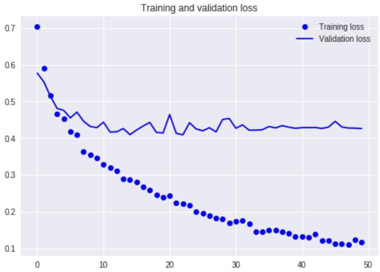
Alfred Sisley

- Exactly the same period.
- Nearly the same style.
- Almost the same scenarios.

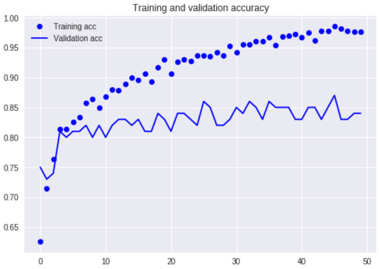
Is it possible?

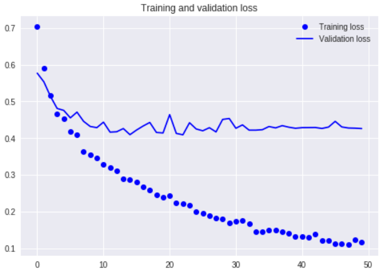
Show me the result...





Show me the result...





Test Accuracy: 86.6%

Future direction:

- Try model ensemble.
- Try batch normalization.
- Try dynamical rate adjustment.
- Try other pre-trained models (tried ResNet, not successful.)

Take-away

- Transfer learning is powerful.
- VGG16 is very easy to train (maybe your first choice).
- Go try the Google colab (too good to be true).
- Keras maybe your first choice, easy and effective.

Learn more about the details and code:

`xingyuzhou.org/blog`