Identify the artist with CNN

Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import json
import os
from tqdm import tqdm, tqdm_notebook
import random

import tensorflow as tf
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.applications import *
from tensorflow.keras.callbacks import *
from tensorflow.keras.initializers import *
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(1)
```

In [2]:
```
print(os.listdir("../input"))
```

```
['artists.csv', 'resized', 'images']
```

In [3]:
```
artists = pd.read_csv('../input/artists.csv')
artists.shape
```

Out[3]:
```
(50, 8)
```

Data Processing

In [4]:
```
# Sort artists by number of paintings
artists = artists.sort_values(by=['paintings'], ascending=False)

# Create a dataframe with artists having more than 200 paintings
artists_top = artists[artists['paintings'] >= 200].reset_index()
artists_top = artists_top[['name', 'paintings']]
#artists_top['class_weight'] = max(artists_top.paintings)/artists_top.paintings
artists_top['class_weight'] = artists_top.paintings.sum() / (artists_top.shape[0] * artists_top.paintings)
artists_top
```

Out[4]:

	name	paintings	class_weight
0	Vincent van Gogh	877	0.445631
1	Edgar Degas	702	0.556721
2	Pablo Picasso	439	0.890246
3	Pierre-Auguste Renoir	336	1.163149
4	Albrecht Dürer	328	1.191519
5	Paul Gauguin	311	1.256650
6	Francisco Goya	291	1.343018
7	Rembrandt	262	1.491672
8	Alfred Sisley	259	1.508951
9	Titian	255	1.532620
10	Marc Chagall	239	1.635223

In [5]:
```
# Set class weights - assign higher weights to underrepresented classes
class_weights = artists_top['class_weight'].to_dict()
class_weights
```

Out[5]:
```
{0: 0.44563076604125634,
 1: 0.5567210567210568,
 2: 0.8902464278318493,
 3: 1.1631493506493507,
 4: 1.1915188470066518,
 5: 1.2566501023092662,
 6: 1.3430178069353327,
 7: 1.491672449687717,
 8: 1.5089505089505089,
 9: 1.532620320855615,
 10: 1.6352225180677062}
```

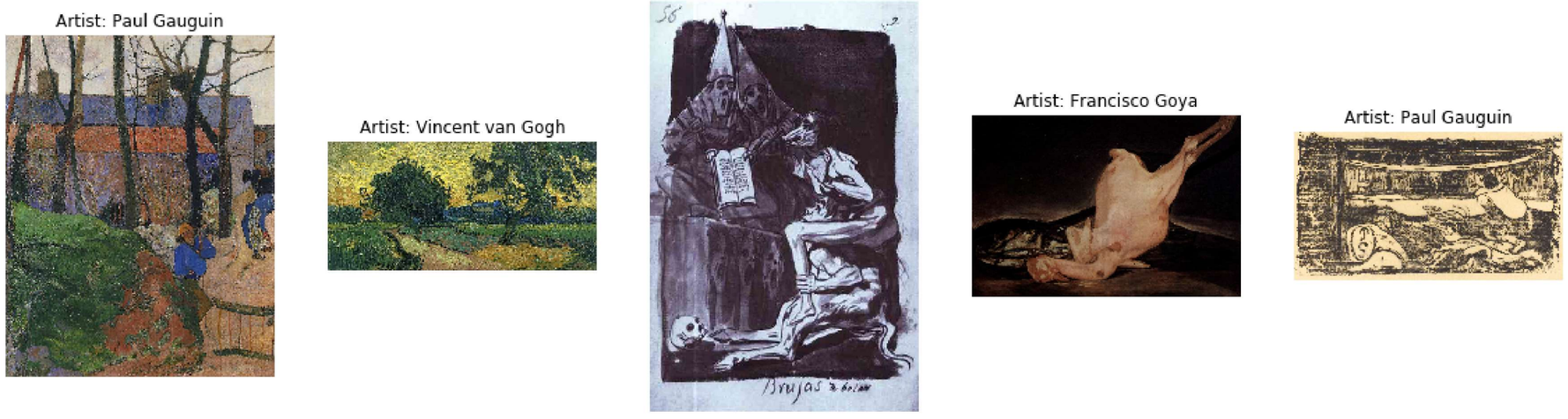
In [6]:
```
# There seems to be an issue recognizing 'Albrecht_Dürer' (the reason is unknown, but it's worth exploring). Therefore, I will update this string as the directory name for 'df's'.
updated_name = "Albrecht_Dürer".replace("_", " ")
artists_top.iloc[4, 0] = updated_name
```

In [7]:
```
# Explore images of top artists
images_dir = '../input/images/images'
artists_dirs = os.listdir(images_dir)
artists_top_name = artists_top['name'].str.replace(' ', '_').values

# See if all directories exist
for name in artists_top_name:
    if os.path.exists(os.path.join(images_dir, name)):
        print("Found -->", os.path.join(images_dir, name))
    else:
        print("Did not find -->", os.path.join(images_dir, name))
```

```
Found --> ../input/images/images/Vincent_van_Gogh
Found --> ../input/images/images/Edgar_Degas
Found --> ../input/images/images/Pablo_Picasso
Found --> ../input/images/images/Pierre-Auguste_Renoir
Found --> ../input/images/images/Albrecht_Dürer
Found --> ../input/images/images/Paul_Gauguin
Found --> ../input/images/images/Francisco_Goya
Found --> ../input/images/images/Rembrandt
Found --> ../input/images/images/Alfred_Sisley
Found --> ../input/images/images/Titian
Found --> ../input/images/images/Marc_Chagall
```

Print few random paintings

In [8]:
```
n = 5
fig, axes = plt.subplots(1, n, figsize=(20,10))

for i in range(n):
    random_artist = random.choice(artists_top_name)
    random_image = random.choice(os.listdir(os.path.join(images_dir, random_artist)))
    random_image_file = os.path.join(images_dir, random_artist, random_image)
    image = plt.imread(random_image_file)
    axes[i].imshow(image)
    axes[i].set_title("Artist: " + random_artist.replace('_', ' '))
    axes[i].axis('off')

plt.show()
```



Data Augmentation

In [9]:
```
# Augment data
batch_size = 16
train_input_shape = (224, 224, 3)
n_classes = artists_top.shape[0]

train_datagen = ImageDataGenerator(validation_split=0.2,
```

```
                                    rescale=1./255.,
                                    #rotation_range=45,
                                    #width_shift_range=0.5,
                                    #height_shift_range=0.5,
                                    shear_range=5,
                                    #zoom_range=0.7,
                                    horizontal_flip=True,
                                    vertical_flip=True,
                                   )

train_generator = train_datagen.flow_from_directory(directory=images_dir,
                                                    class_mode='categorical',
                                                    target_size=train_input_shape[0:2],
                                                    batch_size=batch_size,
                                                    subset="training",
                                                    shuffle=True,
                                                    classes=artists_top_name.tolist()
                                                   )

valid_generator = train_datagen.flow_from_directory(directory=images_dir,
                                                    class_mode='categorical',
                                                    target_size=train_input_shape[0:2],
                                                    batch_size=batch_size,
                                                    subset="validation",
                                                    shuffle=True,
                                                    classes=artists_top_name.tolist()
                                                   )

STEP_SIZE_TRAIN = train_generator.n//train_generator.batch_size
STEP_SIZE_VALID = valid_generator.n//valid_generator.batch_size
print("Total number of batches =", STEP_SIZE_TRAIN, "and", STEP_SIZE_VALID)
```

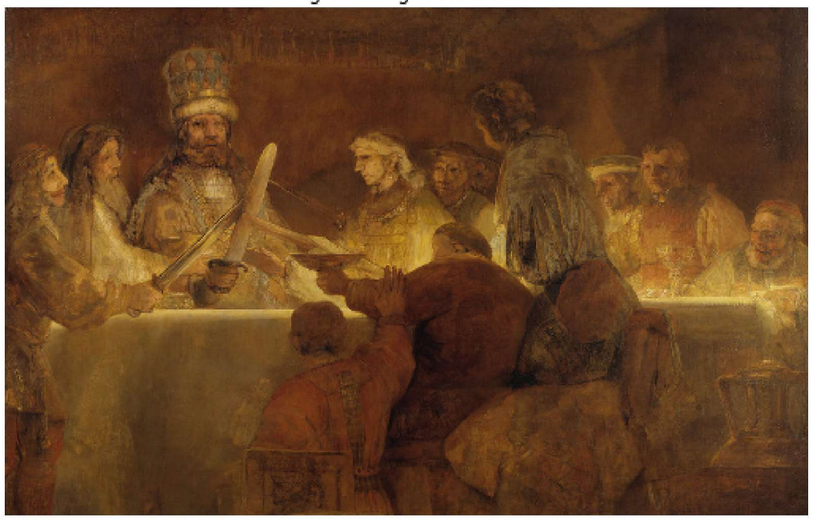
```
Found 3444 images belonging to 11 classes.
Found 855 images belonging to 11 classes.
Total number of batches = 215 and 53
```

Print a random paintings and it's random augmented version

In [10]:

```
# Print a random paintings and it's random augmented version
fig, axes = plt.subplots(1, 2, figsize=(20,10))

random_artist = random.choice(artists_top_name)
random_image = random.choice(os.listdir(os.path.join(images_dir, random_artist)))
random_image_file = os.path.join(images_dir, random_artist, random_image)

# Original image
image = plt.imread(random_image_file)
axes[0].imshow(image)
axes[0].set_title("An original Image of " + random_artist.replace('_', ' '))
axes[0].axis('off')

# Transformed image
aug_image = train_datagen.random_transform(image)
axes[1].imshow(aug_image)
axes[1].set_title("A transformed Image of " + random_artist.replace('_', ' '))
axes[1].axis('off')

plt.show()
```

An original Image of Rembrandt



A transformed Image of Rembrandt

Build Model

In [11]:
```
# Load pre-trained model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=train_input_shape)

for layer in base_model.layers:
    layer.trainable = True
```

```
/opt/conda/lib/python3.6/site-packages/keras_applications/resnet50.py:265: UserWarning: The output shape of `ResNet50(include_top=False)` has been changed since Keras 2.2.0.
  warnings.warn('The output shape of `ResNet50(include_top=False)` '
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.2/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94658560/94653016 [==============================] - 1s 0us/step
```

In [12]:
```
# Add layers at the end
X = base_model.output
X = Flatten()(X)

X = Dense(512, kernel_initializer='he_uniform')(X)
#X = Dropout(0.5)(X)
X = BatchNormalization()(X)
X = Activation('relu')(X)

X = Dense(16, kernel_initializer='he_uniform')(X)
#X = Dropout(0.5)(X)
X = BatchNormalization()(X)
X = Activation('relu')(X)

output = Dense(n_classes, activation='softmax')(X)

model = Model(inputs=base_model.input, outputs=output)
```

In [13]:
```
optimizer = Adam(lr=0.0001)
model.compile(loss='categorical_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])
```

In [14]:
```
n_epoch = 10

early_stop = EarlyStopping(monitor='val_loss', patience=20, verbose=1,
                           mode='auto', restore_best_weights=True)

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5,
                              verbose=1, mode='auto')
```

In [15]:
```
# Train the model - all layers
history1 = model.fit_generator(generator=train_generator, steps_per_epoch=STEP_SIZE_TRAIN,
                               validation_data=valid_generator, validation_steps=STEP_SIZE_VALID,
                               epochs=n_epoch,
                               shuffle=True,
                               verbose=1,
                               callbacks=[reduce_lr],
                               use_multiprocessing=True,
                               workers=16,
                               class_weight=class_weights
                              )
```

```
Epoch 1/10
215/215 [==============================] - 84s 389ms/step - loss: 1.6380 - acc: 0.4772 - val_loss: 4.4245 - val_acc: 0.2040
Epoch 2/10
215/215 [==============================] - 76s 352ms/step - loss: 1.1652 - acc: 0.7412 - val_loss: 2.4852 - val_acc: 0.1781
Epoch 3/10
215/215 [==============================] - 77s 358ms/step - loss: 1.0856 - acc: 0.7520 - val_loss: 2.5971 - val_acc: 0.1415
Epoch 4/10
215/215 [==============================] - 78s 362ms/step - loss: 0.9468 - acc: 0.8188 - val_loss: 2.4894 - val_acc: 0.2394
Epoch 5/10
215/215 [==============================] - 78s 365ms/step - loss: 0.8333 - acc: 0.8518 - val_loss: 1.3618 - val_acc: 0.6380
Epoch 6/10
215/215 [==============================] - 79s 370ms/step - loss: 0.7393 - acc: 0.8775 - val_loss: 1.0604 - val_acc: 0.7866
Epoch 7/10
215/215 [==============================] - 80s 374ms/step - loss: 0.6550 - acc: 0.8938 - val_loss: 0.9928 - val_acc: 0.7783
Epoch 8/10
215/215 [==============================] - 81s 377ms/step - loss: 0.5856 - acc: 0.9081 - val_loss: 0.9164 - val_acc: 0.7866
Epoch 9/10
215/215 [==============================] - 83s 387ms/step - loss: 0.5243 - acc: 0.9128 - val_loss: 0.9072 - val_acc: 0.7995
Epoch 10/10
215/215 [==============================] - 81s 377ms/step - loss: 0.4685 - acc: 0.9285 - val_loss: 0.8422 - val_acc: 0.8208
```

In [16]:
```
# Freeze core ResNet layers and train again
for layer in model.layers:
    layer.trainable = False

for layer in model.layers[:50]:
    layer.trainable = True

optimizer = Adam(lr=0.0001)

model.compile(loss='categorical_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])

n_epoch = 50
history2 = model.fit_generator(generator=train_generator, steps_per_epoch=STEP_SIZE_TRAIN,
                               validation_data=valid_generator, validation_steps=STEP_SIZE_VALID,
                               epochs=n_epoch,
```

```
                                shuffle=True,
                                verbose=1,
                                callbacks=[reduce_lr, early_stop],
                                use_multiprocessing=True,
                                workers=16,
                                class_weight=class_weights
                               )
```

```
Epoch 1/50
215/215 [==============================] - 79s 370ms/step - loss: 0.3794 - acc: 0.9618 - val_loss: 0.8244 - val_acc: 0.8137
Epoch 2/50
215/215 [==============================] - 75s 351ms/step - loss: 0.3965 - acc: 0.9527 - val_loss: 0.8146 - val_acc: 0.8101
Epoch 3/50
215/215 [==============================] - 75s 351ms/step - loss: 0.3791 - acc: 0.9641 - val_loss: 0.7713 - val_acc: 0.8420
Epoch 4/50
215/215 [==============================] - 76s 355ms/step - loss: 0.3827 - acc: 0.9606 - val_loss: 0.8057 - val_acc: 0.8267
Epoch 5/50
215/215 [==============================] - 77s 358ms/step - loss: 0.3898 - acc: 0.9548 - val_loss: 0.7478 - val_acc: 0.8432
Epoch 6/50
215/215 [==============================] - 78s 363ms/step - loss: 0.3668 - acc: 0.9656 - val_loss: 0.7621 - val_acc: 0.8408
Epoch 7/50
215/215 [==============================] - 79s 369ms/step - loss: 0.3602 - acc: 0.9688 - val_loss: 0.7392 - val_acc: 0.8479
Epoch 8/50
215/215 [==============================] - 78s 363ms/step - loss: 0.3749 - acc: 0.9644 - val_loss: 0.7552 - val_acc: 0.8432
Epoch 9/50
215/215 [==============================] - 79s 366ms/step - loss: 0.3726 - acc: 0.9653 - val_loss: 0.7981 - val_acc: 0.8314
Epoch 10/50
215/215 [==============================] - 81s 375ms/step - loss: 0.3729 - acc: 0.9653 - val_loss: 0.7483 - val_acc: 0.8443
Epoch 11/50
215/215 [==============================] - 81s 375ms/step - loss: 0.3742 - acc: 0.9595 - val_loss: 0.7636 - val_acc: 0.8361
Epoch 12/50
214/215 [============================>.] - ETA: 0s - loss: 0.3683 - acc: 0.9683
Epoch 00012: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-06.
215/215 [==============================] - 87s 404ms/step - loss: 0.3682 - acc: 0.9685 - val_loss: 0.8413 - val_acc: 0.8054
Epoch 13/50
215/215 [==============================] - 81s 379ms/step - loss: 0.3639 - acc: 0.9662 - val_loss: 0.7509 - val_acc: 0.8467
Epoch 14/50
215/215 [==============================] - 82s 382ms/step - loss: 0.3527 - acc: 0.9705 - val_loss: 0.7231 - val_acc: 0.8538
Epoch 15/50
215/215 [==============================] - 85s 394ms/step - loss: 0.3597 - acc: 0.9720 - val_loss: 0.7366 - val_acc: 0.8384
Epoch 16/50
215/215 [==============================] - 84s 389ms/step - loss: 0.3473 - acc: 0.9711 - val_loss: 0.7209 - val_acc: 0.8514
Epoch 17/50
215/215 [==============================] - 84s 389ms/step - loss: 0.3652 - acc: 0.9650 - val_loss: 0.7173 - val_acc: 0.8491
Epoch 18/50
215/215 [==============================] - 85s 395ms/step - loss: 0.3497 - acc: 0.9732 - val_loss: 0.7106 - val_acc: 0.8620
Epoch 19/50
215/215 [==============================] - 85s 395ms/step - loss: 0.3570 - acc: 0.9711 - val_loss: 0.7125 - val_acc: 0.8467
Epoch 20/50
215/215 [==============================] - 85s 397ms/step - loss: 0.3586 - acc: 0.9700 - val_loss: 0.7142 - val_acc: 0.8514
Epoch 21/50
215/215 [==============================] - 86s 401ms/step - loss: 0.3578 - acc: 0.9705 - val_loss: 0.7255 - val_acc: 0.8467
Epoch 22/50
215/215 [==============================] - 86s 401ms/step - loss: 0.3520 - acc: 0.9717 - val_loss: 0.7142 - val_acc: 0.8491
Epoch 23/50
215/215 [==============================] - 87s 403ms/step - loss: 0.3580 - acc: 0.9705 - val_loss: 0.7100 - val_acc: 0.8491
Epoch 24/50
215/215 [==============================] - 89s 415ms/step - loss: 0.3532 - acc: 0.9758 - val_loss: 0.7169 - val_acc: 0.8550
Epoch 25/50
215/215 [==============================] - 89s 413ms/step - loss: 0.3498 - acc: 0.9749 - val_loss: 0.7299 - val_acc: 0.8502
Epoch 26/50
215/215 [==============================] - 89s 414ms/step - loss: 0.3619 - acc: 0.9705 - val_loss: 0.7189 - val_acc: 0.8526
Epoch 27/50
215/215 [==============================] - 89s 415ms/step - loss: 0.3520 - acc: 0.9717 - val_loss: 0.7198 - val_acc: 0.8502
Epoch 28/50
214/215 [============================>.] - ETA: 0s - loss: 0.3604 - acc: 0.9678
Epoch 00028: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-07.
215/215 [==============================] - 89s 415ms/step - loss: 0.3598 - acc: 0.9679 - val_loss: 0.7432 - val_acc: 0.8455
```

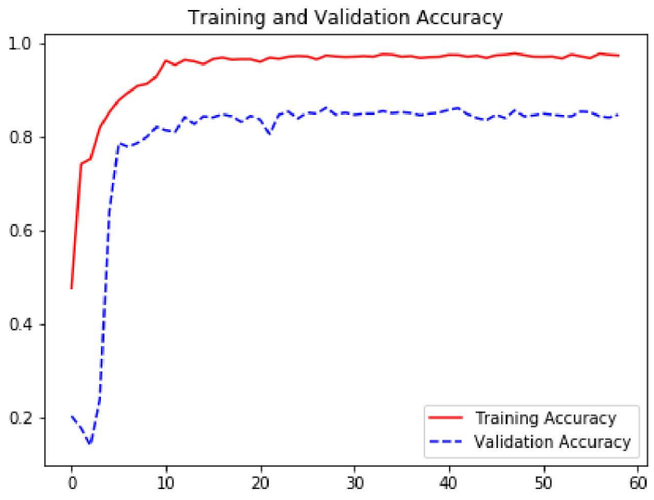
```
Epoch 29/50
215/215 [==============================] - 91s 421ms/step - loss: 0.3640 - acc: 0.9697 - val_loss: 0.7052 - val_acc: 0.8491
Epoch 30/50
215/215 [==============================] - 91s 424ms/step - loss: 0.3560 - acc: 0.9702 - val_loss: 0.7064 - val_acc: 0.8514
Epoch 31/50
215/215 [==============================] - 91s 425ms/step - loss: 0.3480 - acc: 0.9743 - val_loss: 0.7057 - val_acc: 0.8573
Epoch 32/50
215/215 [==============================] - 92s 430ms/step - loss: 0.3492 - acc: 0.9740 - val_loss: 0.7085 - val_acc: 0.8608
Epoch 33/50
215/215 [==============================] - 93s 433ms/step - loss: 0.3501 - acc: 0.9705 - val_loss: 0.7235 - val_acc: 0.8479
Epoch 34/50
214/215 [============================>.] - ETA: 0s - loss: 0.3633 - acc: 0.9725
Epoch 00034: ReduceLROnPlateau reducing learning rate to 9.999999974752428e-08.
215/215 [==============================] - 94s 437ms/step - loss: 0.3631 - acc: 0.9726 - val_loss: 0.7370 - val_acc: 0.8396
Epoch 35/50
215/215 [==============================] - 97s 451ms/step - loss: 0.3653 - acc: 0.9679 - val_loss: 0.7452 - val_acc: 0.8349
Epoch 36/50
215/215 [==============================] - 94s 439ms/step - loss: 0.3476 - acc: 0.9735 - val_loss: 0.7269 - val_acc: 0.8467
Epoch 37/50
215/215 [==============================] - 96s 446ms/step - loss: 0.3531 - acc: 0.9746 - val_loss: 0.7329 - val_acc: 0.8396
Epoch 38/50
215/215 [==============================] - 96s 446ms/step - loss: 0.3459 - acc: 0.9775 - val_loss: 0.7159 - val_acc: 0.8561
Epoch 39/50
214/215 [============================>.] - ETA: 0s - loss: 0.3505 - acc: 0.9737
Epoch 00039: ReduceLROnPlateau reducing learning rate to 1.0000000116860975e-08.
215/215 [==============================] - 97s 453ms/step - loss: 0.3506 - acc: 0.9738 - val_loss: 0.7296 - val_acc: 0.8432
Epoch 40/50
215/215 [==============================] - 98s 454ms/step - loss: 0.3498 - acc: 0.9702 - val_loss: 0.7188 - val_acc: 0.8491
Epoch 42/50
215/215 [==============================] - 99s 461ms/step - loss: 0.3579 - acc: 0.9708 - val_loss: 0.7179 - val_acc: 0.8467
Epoch 43/50
215/215 [==============================] - 101s 467ms/step - loss: 0.3549 - acc: 0.9666 - val_loss: 0.7307 - val_acc: 0.8443
Epoch 44/50
214/215 [============================>.] - ETA: 0s - loss: 0.3603 - acc: 0.9750
Epoch 00044: ReduceLROnPlateau reducing learning rate to 9.999999939225292e-10.
215/215 [==============================] - 99s 461ms/step - loss: 0.3600 - acc: 0.9751 - val_loss: 0.7201 - val_acc: 0.8432
Epoch 45/50
215/215 [==============================] - 101s 468ms/step - loss: 0.3512 - acc: 0.9714 - val_loss: 0.7092 - val_acc: 0.8538
Epoch 46/50
215/215 [==============================] - 101s 470ms/step - loss: 0.3638 - acc: 0.9673 - val_loss: 0.7137 - val_acc: 0.8526
Epoch 47/50
215/215 [==============================] - 102s 474ms/step - loss: 0.3469 - acc: 0.9770 - val_loss: 0.7203 - val_acc: 0.8432
Epoch 48/50
155/215 [====================>.........] - ETA: 22s - loss: 0.3464 - acc: 0.9756
Epoch 00049: ReduceLROnPlateau reducing learning rate to 9.999999717180686e-11.
Restoring model weights from the end of the best epoch.
215/215 [==============================] - 116s 540ms/step - loss: 0.3498 - acc: 0.9732 - val_loss: 0.7195 - val_acc: 0.8467
Epoch 00049: early stopping
```

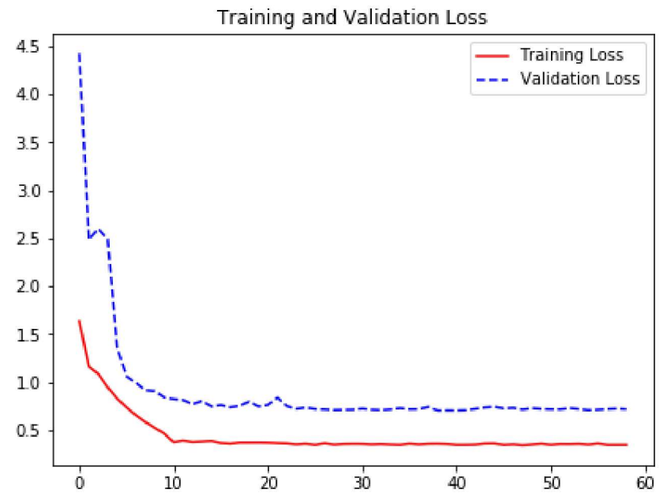
Training graph

In [17]:

```python
# Merge history1 and history2
history = {}
history['loss'] = history1.history['loss'] + history2.history['loss']
history['acc'] = history1.history['acc'] + history2.history['acc']
history['val_loss'] = history1.history['val_loss'] + history2.history['val_loss']
history['val_acc'] = history1.history['val_acc'] + history2.history['val_acc']
history['lr'] = history1.history['lr'] + history2.history['lr']
```

In [18]:
```
# Plot the training graph
def plot_training(history):
    acc = history['acc']
    val_acc = history['val_acc']
    loss = history['loss']
    val_loss = history['val_loss']
    epochs = range(len(acc))

    fig, axes = plt.subplots(1, 2, figsize=(15,5))

    axes[0].plot(epochs, acc, 'r-', label='Training Accuracy')
    axes[0].plot(epochs, val_acc, 'b--', label='Validation Accuracy')
    axes[0].set_title('Training and Validation Accuracy')
    axes[0].legend(loc='best')

    axes[1].plot(epochs, loss, 'r-', label='Training Loss')
    axes[1].plot(epochs, val_loss, 'b--', label='Validation Loss')
    axes[1].set_title('Training and Validation Loss')
    axes[1].legend(loc='best')

    plt.show()

plot_training(history)
```





Evaluate performance

In [19]:
```
# Prediction accuracy on train data
score = model.evaluate_generator(train_generator, verbose=1)
print("Prediction accuracy on train data =", score[1])
```

```
216/216 [==============================] - 93s 433ms/step - loss: 0.3012 - acc: 0.9925
Prediction accuracy on train data = 0.99245065
```

In [20]:
```
# Prediction accuracy on CV data
score = model.evaluate_generator(valid_generator, verbose=1)
print("Prediction accuracy on CV data =", score[1])
```

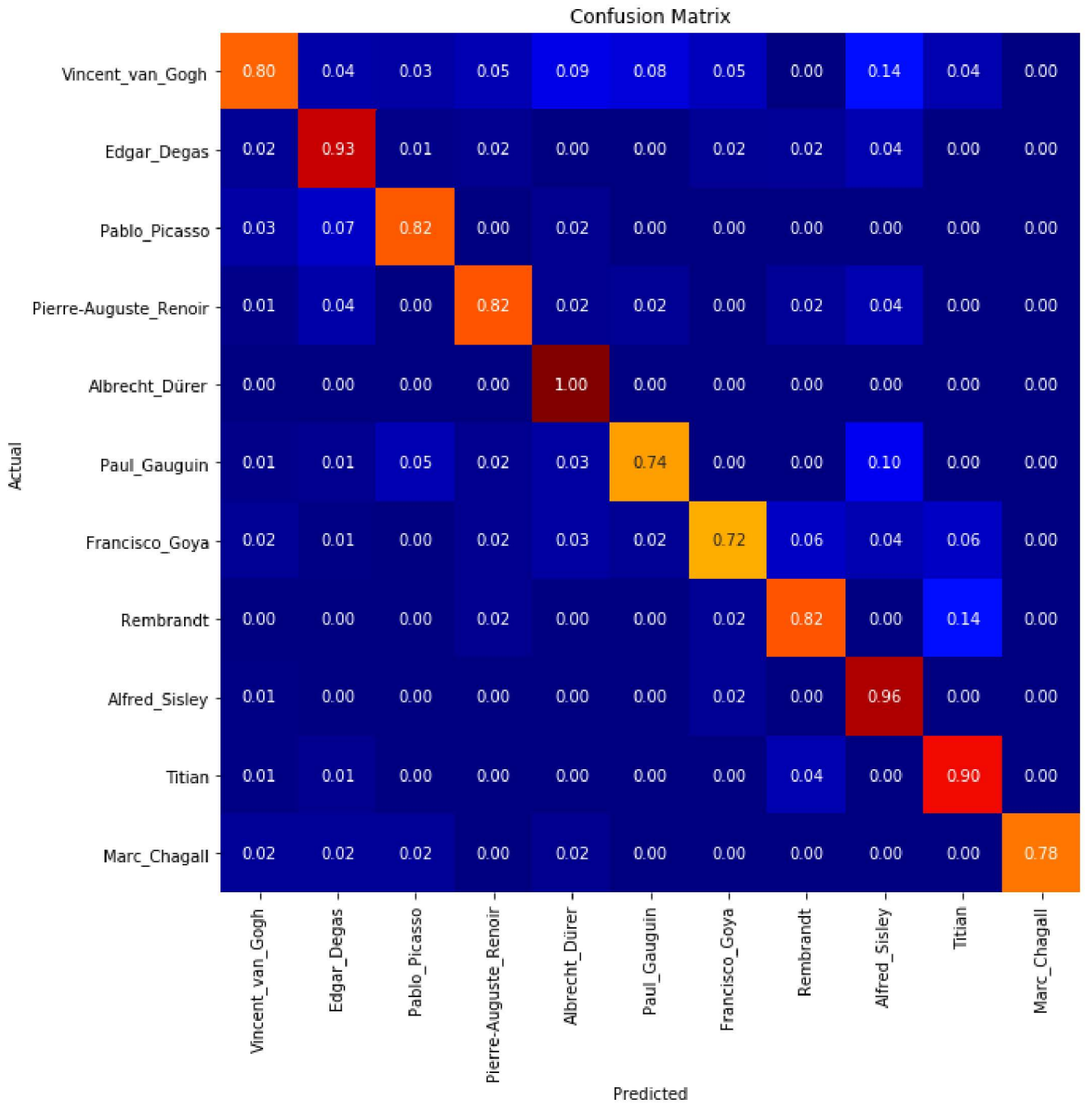
```
54/54 [==============================] - 23s 425ms/step - loss: 0.7159 - acc: 0.8538
Prediction accuracy on CV data = 0.8538012
```

Confusion Matrix.

Look at the style of the artists which the model thinks are almost similar.

In [21]:

```
# Classification report and confusion matrix
from sklearn.metrics import *
import seaborn as sns

tick_labels = artists_top_name.tolist()

def showClassficationReport_Generator(model, valid_generator, STEP_SIZE_VALID):
    # Loop on each generator batch and predict
    y_pred, y_true = [], []
    for i in range(STEP_SIZE_VALID):
        (X,y) = next(valid_generator)
        y_pred.append(model.predict(X))
        y_true.append(y)

    # Create a flat list for y_true and y_pred
    y_pred = [subresult for result in y_pred for subresult in result]
    y_true = [subresult for result in y_true for subresult in result]

    # Update Truth vector based on argmax
    y_true = np.argmax(y_true, axis=1)
    y_true = np.asarray(y_true).ravel()

    # Update Prediction vector based on argmax
    y_pred = np.argmax(y_pred, axis=1)
    y_pred = np.asarray(y_pred).ravel()

    # Confusion Matrix
    fig, ax = plt.subplots(figsize=(10,10))
    conf_matrix = confusion_matrix(y_true, y_pred, labels=np.arange(n_classes))
    conf_matrix = conf_matrix/np.sum(conf_matrix, axis=1)
    sns.heatmap(conf_matrix, annot=True, fmt=".2f", square=True, cbar=False,
                cmap=plt.cm.jet, xticklabels=tick_labels, yticklabels=tick_labels,
                ax=ax)
    ax.set_ylabel('Actual')
    ax.set_xlabel('Predicted')
    ax.set_title('Confusion Matrix')
    plt.show()

    print('Classification Report:')
    print(classification_report(y_true, y_pred, labels=np.arange(n_classes), target_names=artists_top_name.tolist()))

showClassficationReport_Generator(model, valid_generator, STEP_SIZE_VALID)
```


Confusion Matrix
Vincent_van_Gogh 0.80 0.04 0.03 0.05 0.09 0.08 0.05 0.00 0.14 0.04 0.00
Edgar_Degas 0.02 0.93 0.01 0.02 0.00 0.00 0.02 0.02 0.04 0.00 0.00
Pablo_Picasso 0.03 0.07 0.82 0.00 0.02 0.00 0.00 0.00 0.00 0.00 0.00
Pierre-Auguste_Renoir 0.01 0.04 0.00 0.82 0.02 0.02 0.00 0.02 0.04 0.00 0.00
Albrecht_Dürer 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
Paul_Gauguin 0.01 0.01 0.05 0.02 0.03 0.74 0.00 0.00 0.10 0.00 0.00
Francisco_Goya 0.02 0.01 0.00 0.02 0.03 0.02 0.72 0.06 0.04 0.06 0.00
Rembrandt 0.00 0.00 0.00 0.02 0.00 0.00 0.02 0.82 0.00 0.14 0.00
Alfred_Sisley 0.01 0.00 0.00 0.00 0.00 0.00 0.02 0.00 0.96 0.00 0.00
Titian 0.01 0.01 0.00 0.00 0.00 0.00 0.00 0.04 0.00 0.90 0.00
Marc_Chagall 0.02 0.02 0.02 0.00 0.02 0.00 0.00 0.00 0.00 0.00 0.78
Actual
Vincent_van_Gogh
Edgar_Degas
Pablo_Picasso
Pierre-Auguste_Renoir
Albrecht_Dürer
Paul_Gauguin
Francisco_Goya
Rembrandt
Alfred_Sisley
Titian
Marc_Chagall
Predicted

```
Classification Report:
                      precision    recall  f1-score   support

     Vincent_van_Gogh       0.86      0.80      0.83       174
         Edgar_Degas        0.83      0.93      0.88       138
       Pablo_Picasso        0.88      0.82      0.85        87
Pierre-Auguste_Renoir       0.89      0.82      0.85        66
      Albrecht_Dürer       0.83      1.00      0.91        65
        Paul_Gauguin       0.87      0.74      0.80        62
       Francisco_Goya       0.88      0.72      0.79        58
           Rembrandt        0.85      0.82      0.84        50
       Alfred_Sisley        0.73      0.96      0.83        51
              Titian        0.79      0.90      0.84        51
        Marc_Chagall        1.00      0.78      0.88        46

            accuracy                           0.85       848
           macro avg        0.86      0.85      0.85       848
        weighted avg        0.85      0.85      0.85       848
```

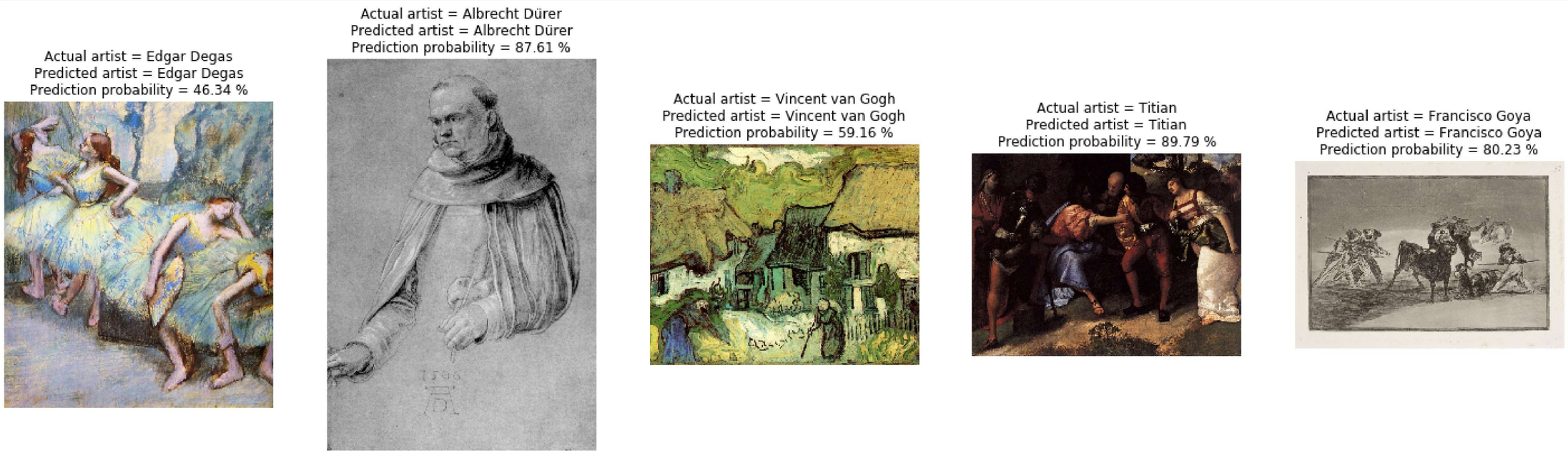
Evaluate performance by predicting on random images from dataset

In [22]:

```
# Prediction
from keras.preprocessing import *

n = 5
fig, axes = plt.subplots(1, n, figsize=(25,10))

for i in range(n):
random_artist = random.choice(artists_top_name)
random_image = random.choice(os.listdir(os.path.join(images_dir, random_artist)))
random_image_file = os.path.join(images_dir, random_artist, random_image)

# Original image

test_image = image.load_img(random_image_file, target_size=(train_input_shape[0:2]))

# Predict artist
test_image = image.img_to_array(test_image)
test_image /= 255.
test_image = np.expand_dims(test_image, axis=0)

prediction = model.predict(test_image)
prediction_probability = np.amax(prediction)
prediction_idx = np.argmax(prediction)

labels = train_generator.class_indices
labels = dict((v,k) for k,v in labels.items())

#print("Actual artist =", random_artist.replace('_', ' '))
#print("Predicted artist =", labels[prediction_idx].replace('_', ' '))
#print("Prediction probability =", prediction_probability*100, "%")

title = "Actual artist = {}\nPredicted artist = {}\nPrediction probability = {:.2f} %" \
            .format(random_artist.replace('_', ' '), labels[prediction_idx].replace('_', ' '),
                    prediction_probability*100)

# Print image
axes[i].imshow(plt.imread(random_image_file))
```

```
    axes[i].set_title(title)
    axes[i].axis('off')

plt.show()
```

```
Using TensorFlow backend.
```



In [23]:
```
# Predict from web - this is an image of Titian.
# Replace 'url' with any image of one of the 11 artists above and run this cell.
url = 'https://www.gpsmycity.com/img/gd/2081.jpg'

import imageio
import cv2

web_image = imageio.imread(url)
web_image = cv2.resize(web_image, dsize=train_input_shape[0:2], )
web_image = image.img_to_array(web_image)
web_image /= 255.
web_image = np.expand_dims(web_image, axis=0)


prediction = model.predict(web_image)
prediction_probability = np.amax(prediction)
prediction_idx = np.argmax(prediction)

print("Predicted artist =", labels[prediction_idx].replace('_', ' '))
print("Prediction probability =", prediction_probability*100, "%")

plt.imshow(imageio.imread(url))
plt.axis('off')
plt.show()
```

```
Predicted artist = Titian
Prediction probability = 89.85584378242493 %
```