INFO7374 Final Project Report

Summary	This codelab is the final project report
URL	https://github.com/ll1195831146/Infor7374-AI/tree/master/Final%20Project
Category	Photo tagging
Environment	Keras, Python
Status	Done
Feedback Link	https://github.com/ll1195831146/Infor7374-AI/tree/master/Final%20Project/issues
Author	Yuchen He, Lei Liu, Xiangyu Chen

Kaggle Competition

--iMet Collection 2019

04.26.2019

Competition URL: https://www.kaggle.com/c/imet-2019-fgvc6

Xiangyu Chen
Lei Liu
Yuchen He

Final Project : INFO 7374 Special Topics in Info Systems

Project Goal

To describe the object from an art history perspective and add fine-grained attributes to aid in the visual understanding of objects in the The Metropolitan Museum of Art.
(Image multi-label classification)

Implementation details

Limitation

9 hour runtime maximum.

EDA

Dataset:

Labels.csv contains culture & tag id and attribute name pairs.
Train.csv contains image id and corresponding attribute ids.
Test.zip and train.zip contains image files.

- input (read-only)
  - iMet Collection 2019 - FGV...
    - labels.csv 1103 x 2
    - sample_submission.csv 7443 x 2
    - test.zip
    - train.csv 109k x 2
    - train.zip

```
labels_df.head()
```

	attribute_id	attribute_name
0	0	culture::abruzzi
1	1	culture::achaemenid
2	2	culture::aegean
3	3	culture::afghan
4	4	culture::after british

```
labels_df.tail()
```

	attribute_id	attribute_name
1098	1098	tag::writing implements
1099	1099	tag::writing systems
1100	1100	tag::zeus
1101	1101	tag::zigzag pattern
1102	1102	tag::zodiac

train.csv

```python
train_df = pd.read_csv("../input/train.csv")
train_df.head()
```

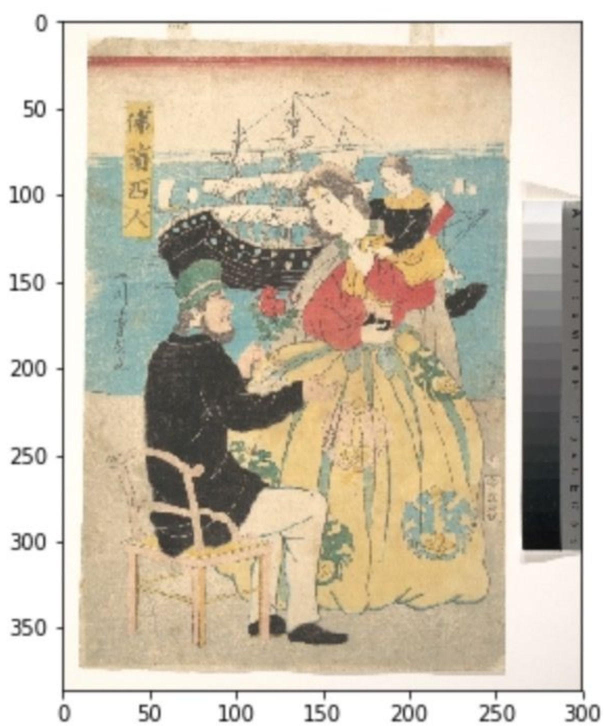
	id	attribute_ids
0	1000483014d91860	147 616 813
1	1000fe2e667721fe	51 616 734 813
2	1001614cb89646ee	776
3	10041eb49b297c08	51 671 698 813 1092
4	100501c227f8beea	13 404 492 903 1093

Number of Tags:

Culture: 398
Tag: 705
Unknown: 0
Total: 1103

Example:

This is a example for a image with attributes. This image has 6 attributes according to its content , 1 culture and 5 tags.



culture::japan
tag::chairs
tag::girls
tag::men
tag::ships
tag::women

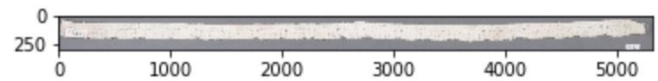
Weird Images:

There are a lot of weird images in train and test. Some images have very huge height or width. For these images, we cannot resize them into standard format directly.

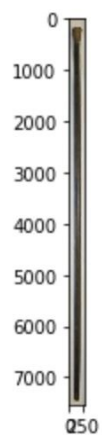
Max height: 7351.0 Min height: 300.0

Max width: 5314.0 Min width: 300.0

```
img = np.asarray(Image.open(str(train_max_width[1])))
plt.imshow(img)
plt.show()
```




```
img = np.asarray(Image.open(str(train_max_height[1])))
plt.imshow(img)
plt.show()
```



After we scale it up, we have a weird image and cannot recognize what it is:

```
resized_img = cv2.resize(img, (256, 256))
resized_img_pil = Image.fromarray(resized_img)
resized_img_pil
```

Resize weird Images:

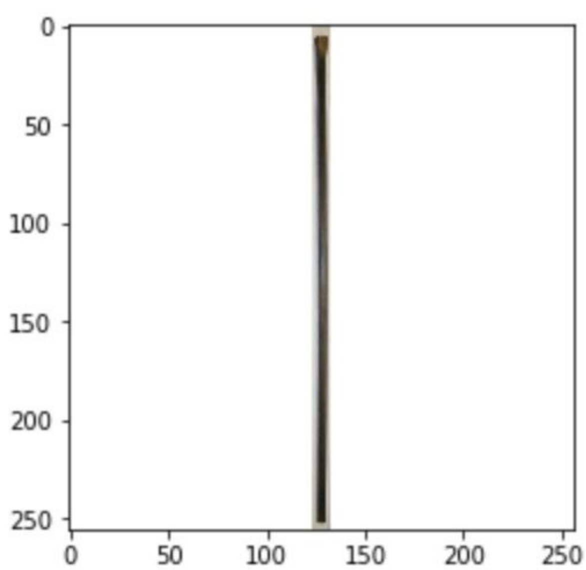
To avoid weird images, we create a function to pad narrow edge to make images look like in a standard size.

We set the aspect ratio threshold as 0.5, the images with aspect ratio greater than 0.5(height/width > 0.5 or width/height > 0.5) will be padded.

```
def resize_weird_images(img, SIZE):
    aspect_ratio = img.size[0] / img.size[1]
    if aspect_ratio < 0.5:
        w_resized = int(img.size[0] * SIZE / img.size[1])
        resized = img.resize((w_resized ,SIZE))
        pad_width = SIZE - w_resized
        padding = (pad_width // 2, 0, pad_width-(pad_width//2), 0)
        img = ImageOps.expand(resized, padding, fill="white")
    elif aspect_ratio > 2:
        h_resized = int(img.size[1] * SIZE / img.size[0])
        resized = img.resize((SIZE, h_resized))
        pad_height = SIZE - h_resized
        padding = (0, pad_height // 2, 0,
pad_height-(pad_height//2))
        img = ImageOps.expand(resized, padding, fill="white")

    return img
```

Pad and resize image to (256, 256) :

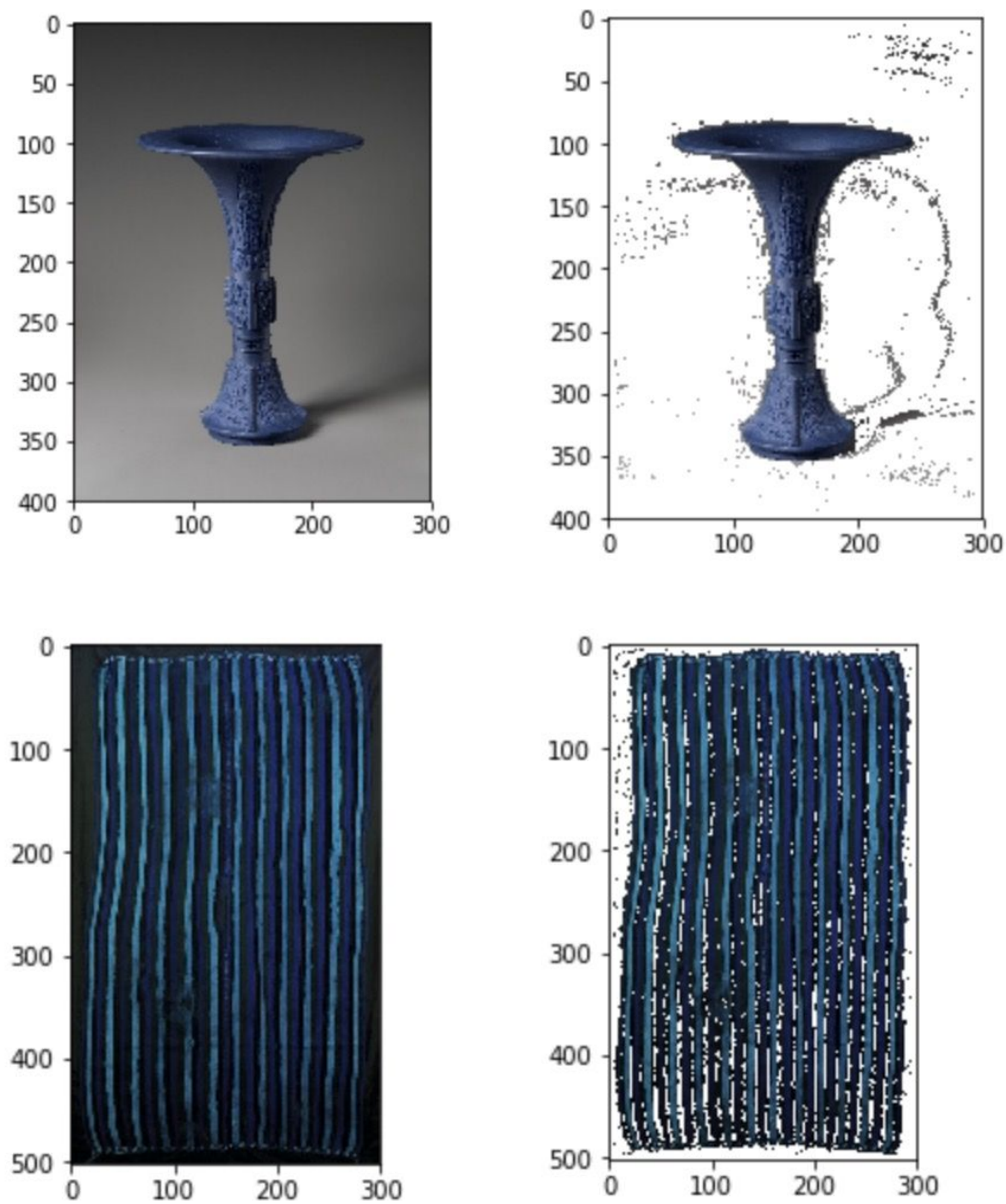


Experiments:

Then we experimented several methods to improve model accuracy.

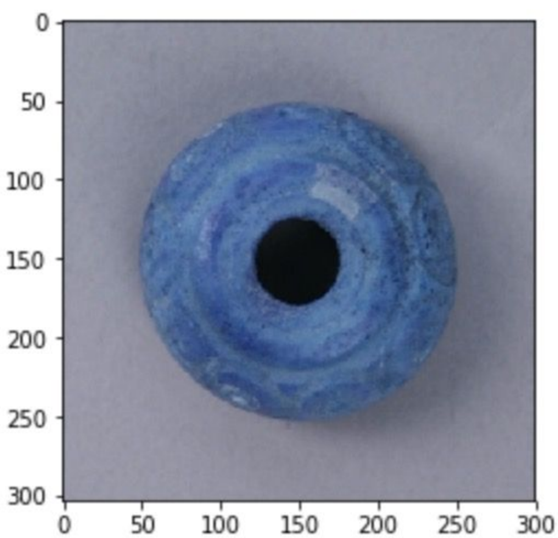
Remove backgrounds:

Training and testing images have different background colors. So removing backgrounds can remove the effect of background on image recognition. But it may also remove some features of image itself. Through our experiments, removing backgrounds will cost much time and not improve the accuracy.

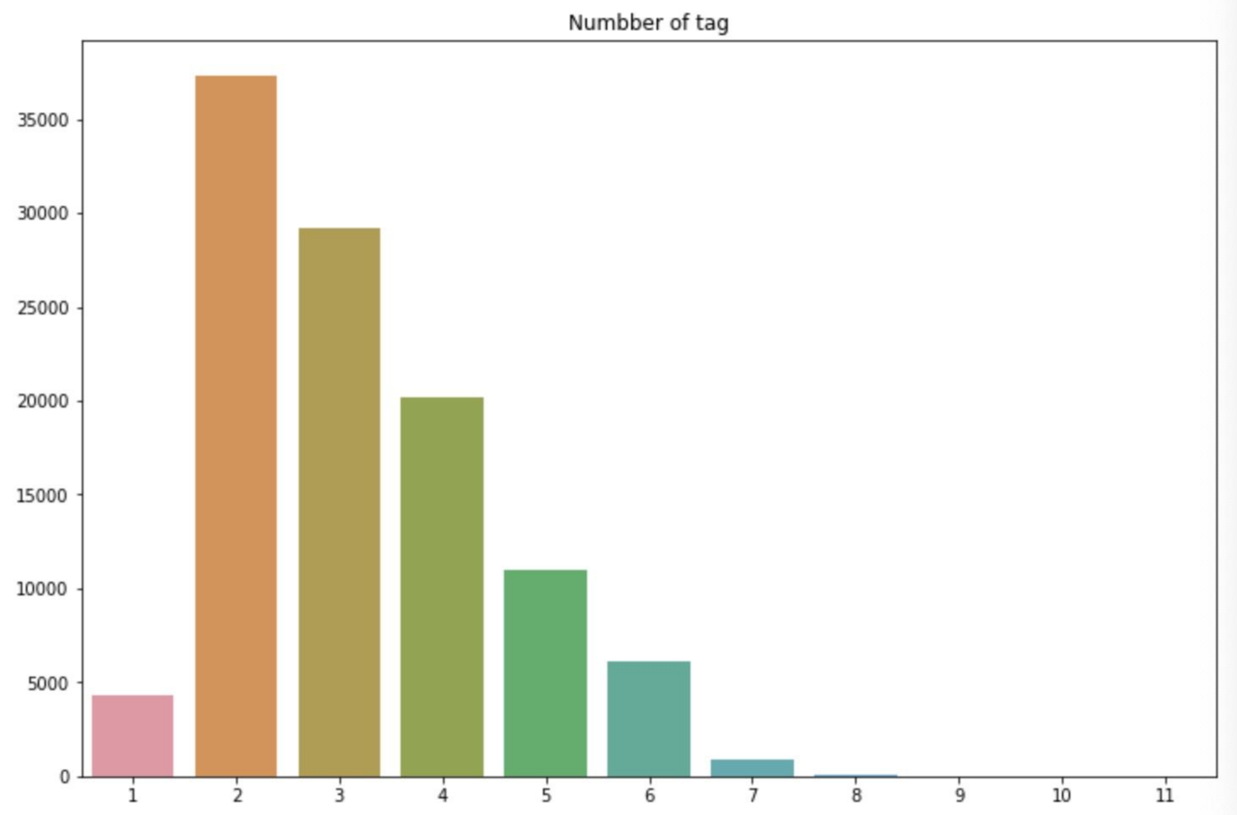


Extract contours:

We also experimented on extracting image contours. But the results show that it cannot help improve the model accuracy.



Loss Function:



Focal Loss:

The focal loss is designed to address class imbalance by down-weighting inliers such that their contribution to the total loss is small even if their number is large. It focuses on training a sparse set of hard examples.

$$\mathrm{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t).$$

```
epsilon = K.epsilon()
def focal_loss(y_true, y_pred):
    pt = y_pred * y_true + (1-y_pred) * (1-y_true)
    pt = K.clip(pt, epsilon, 1-epsilon)
    CE = -K.log(pt)
    FL = K.pow(1-pt, 2.0) * CE
    loss = K.sum(FL, axis=1)
    return loss
```

Analysis of models

After EDA, we created many models to find the best performance in limited time.

Same Parameters:
Image size: 256
Learning rate: 1e-4
NUM_CLASSES = 1103

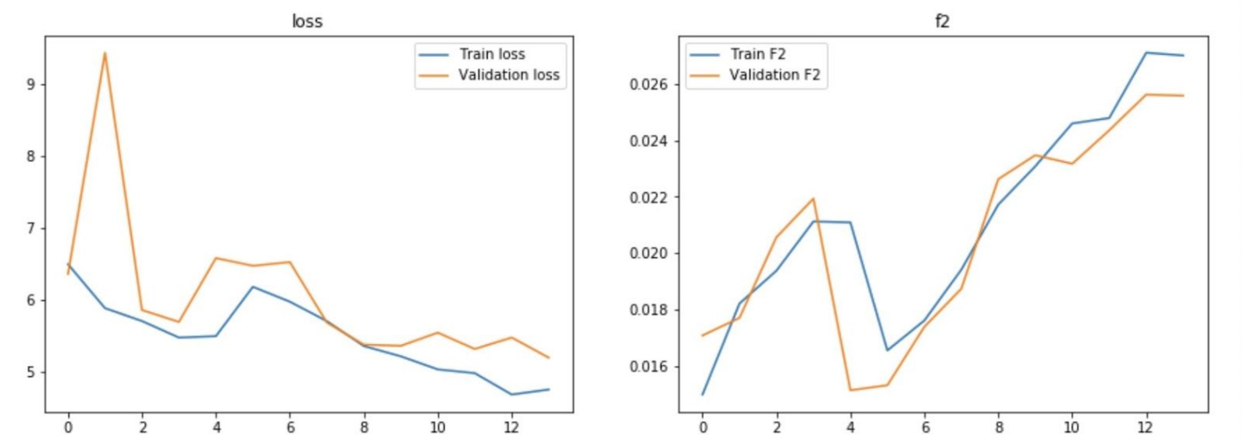
Results From Progress Report

bacth_size: 256

Pre-trained Model Name	Loss: Focal	F2 Score
ResNet-50	5.5789	0.202
InceptionResNetV2	3.9787	0.367
Xception	3.9788	0.358
Vgg19	5.1814	0.075

Final Model and Results

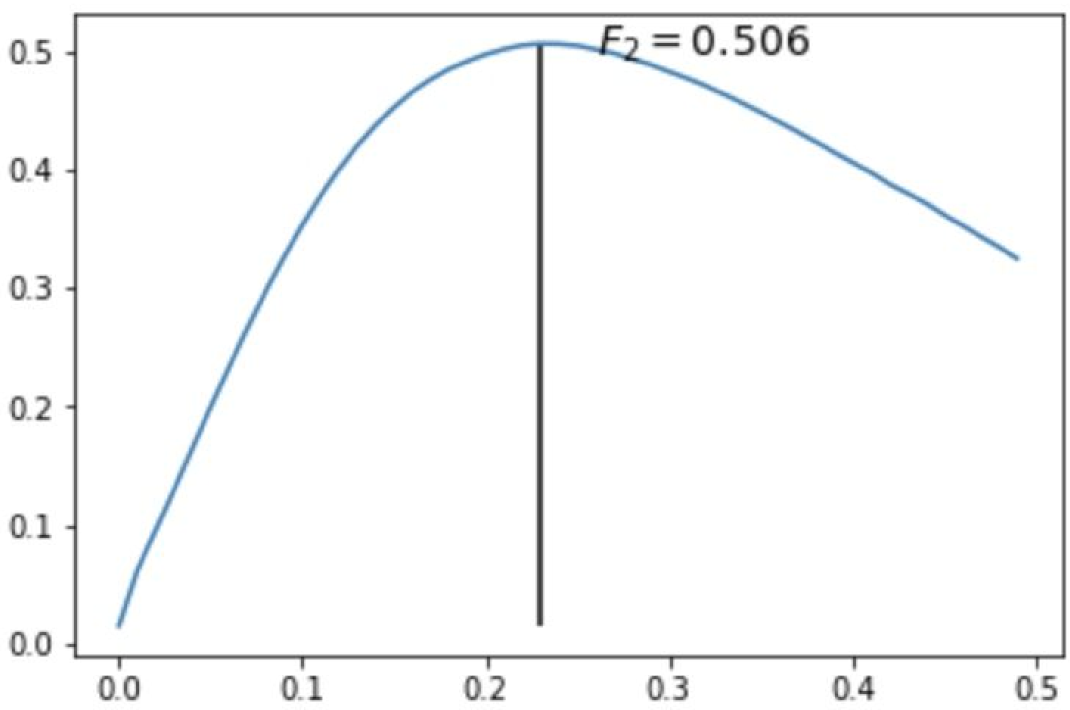
MODEL NAME	EPOCHS	BATCH_SIZE	LOSS	F2-score
Xception + binary loss	25	64	0.007	0.547
InceptionResNetV2 + focal loss	7	64	3.9787	0.534
ResNet-50 + binary loss	24	64	0.0106	0.532
InceptionResNetV2 + Attention + focal loss	14	64	5.1979	0.506
ResNetV2 + Attention + binary loss	14	64	0.009	0.501
InceptionResNetV2 + Attention + focal loss	14	64	5.7402	0.451
Vgg19 + Attention + focal loss	7	256	3.9415	0.348
ResNetV2 + Attention + focal loss + contour	14	64	6.9237	0.338




```
best_thr, best_score = find_best_fixed_threshold(
```

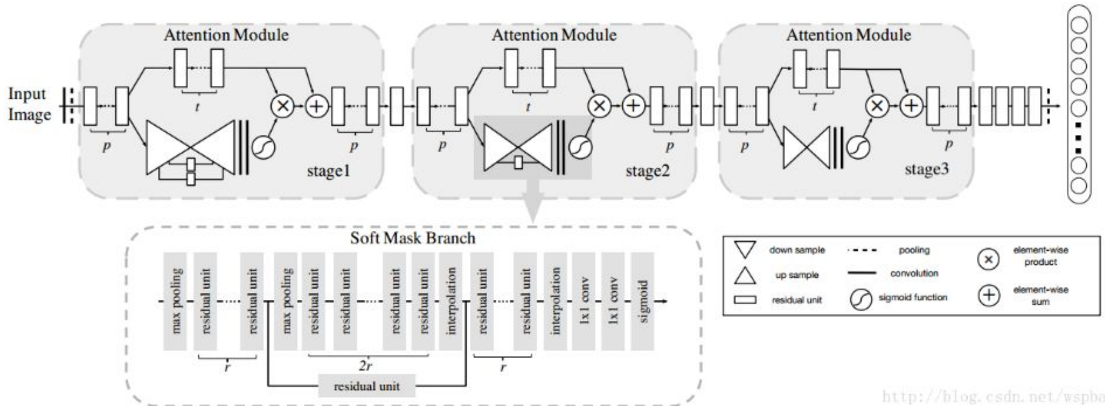
```
100%|██████████| 50/50 [00:22<00:00,  2.24it/s]
```

```
thr=0.230 F2=0.506
```



Details on Attention Layer

Residual Attention Network for Image Classification



Layer	Output Size	Attention-56	Attention-92
Conv1	112×112	7 × 7, 64, stride 2	
Max pooling	56×56	3 × 3 stride 2	
Residual Unit	56×56	$\begin{pmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{pmatrix} \times 1$	
Attention Module	56×56	Attention ×1	Attention ×1
Residual Unit	28×28	$\begin{pmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{pmatrix} \times 1$	
Attention Module	28×28	Attention ×1	Attention ×2
Residual Unit	14×14	$\begin{pmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{pmatrix} \times 1$	
Attention Module	14×14	Attention ×1	Attention ×3
Residual Unit	7×7	$\begin{pmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{pmatrix} \times 3$	
Average pooling	1×1	7 × 7 stride 1	
FC,Softmax	1000		
params$\times10^6$		31.9	51.3
FLOPs$\times10^9$		6.2	10.4
Trunk depth		56	92

https://arxiv.org/pdf/1704.06904.pdf

```
def Attention(X, filters, base):
    F1, F2, F3 = filters
    name_base = base

    X = res_identity(X, filters, name_base+ '/Pre_Residual_id')
    X_Trunk = Trunk_block(X, filters, name_base+ '/Trunk')
    X = MaxPooling2D((3,3), strides=(2,2), padding='same',
name=name_base+ '/Mask/pool_3')(X)
    X = res_identity(X, filters, name_base+
'/Mask/Residual_id_3_Down')
    Residual_id_3_Down_shortcut = X
    Residual_id_3_Down_branched = res_identity(X, filters,
name_base+ '/Mask/Residual_id_3_Down_branched')
    X = MaxPooling2D((3,3), strides=(2,2), padding='same',
name=name_base+ '/Mask/pool_2')(X)
    X = res_identity(X, filters, name_base+
'/Mask/Residual_id_2_Down')
    Residual_id_2_Down_shortcut = X
    Residual_id_2_Down_branched = res_identity(X, filters,
name_base+ '/Mask/Residual_id_2_Down_branched')
    X = MaxPooling2D((3,3), strides=(2,2), padding='same',
name=name_base+ '/Mask/pool_1')(X)
    X = res_identity(X, filters, name_base+
'/Mask/Residual_id_1_Down')
    X = res_identity(X, filters, name_base+
'/Mask/Residual_id_1_Up')
    temp_name1 = name_base+ "/Mask/Interpool_1"
    X = Lambda(interpolation, arguments={'ref_tensor':
Residual_id_2_Down_shortcut,'name':temp_name1})(X)
    X = Add(name=base + '/Mask/Add_after_Interpool_1')([X,
Residual_id_2_Down_branched])
```

```
    X = res_identity(X, filters, name_base+
'/Mask/Residual_id_2_Up')

    temp_name2 = name_base+ "/Mask/Interpool_2"

    X = Lambda(interpolation, arguments={'ref_tensor':
Residual_id_3_Down_shortcut,'name':temp_name2})(X)

    X = Add(name=base + '/Mask/Add_after_Interpool_2')([X,
Residual_id_3_Down_branched])

    X = res_identity(X, filters, name_base+
'/Mask/Residual_id_3_Up')

    temp_name3 = name_base+ "/Mask/Interpool_3"

    X = Lambda(interpolation, arguments={'ref_tensor':
X_Trunk,'name':temp_name3})(X)

    X = BatchNormalization(axis=-1, name=name_base +
'/Mask/Interpool_3/bn_1')(X)

    X = Activation('relu', name=name_base +
'/Mask/Interpool_3/relu_1')(X)

    X = Conv2D(F3, kernel_size=(1,1), strides=(1,1),
padding='valid', name=name_base + '/Mask/Interpool_3/conv_1',
kernel_initializer=glorot_uniform(seed=0))(X)

    X = BatchNormalization(axis=-1, name=name_base +
'/Mask/Interpool_3/bn_2')(X)

    X = Activation('relu', name=name_base +
'/Mask/Interpool_3/relu_2')(X)

    X = Conv2D(F3, kernel_size=(1,1), strides=(1,1),
padding='valid', name=name_base + '/Mask/Interpool_3/conv_2',
kernel_initializer=glorot_uniform(seed=0))(X)

    X = Activation('sigmoid', name=name_base+'/Mask/sigmoid')(X)

    X = Multiply(name=name_base+'/Mutiply')([X_Trunk,X])

    X = Add(name=name_base+'/Add')([X_Trunk,X])

    X = res_identity(X, filters, name_base+ '/Post_Residual_id')

    return X
```