Comparison of Class Activation Maps (CAMs) for CNN and Hybrid Models Performing Image Classification based on Fashion MNIST Dataset

Presented by Anuva Suresh and Bhanu Kaushik

Introduction

Image Classification

↓

Fashion Industry

↓

e-commerce

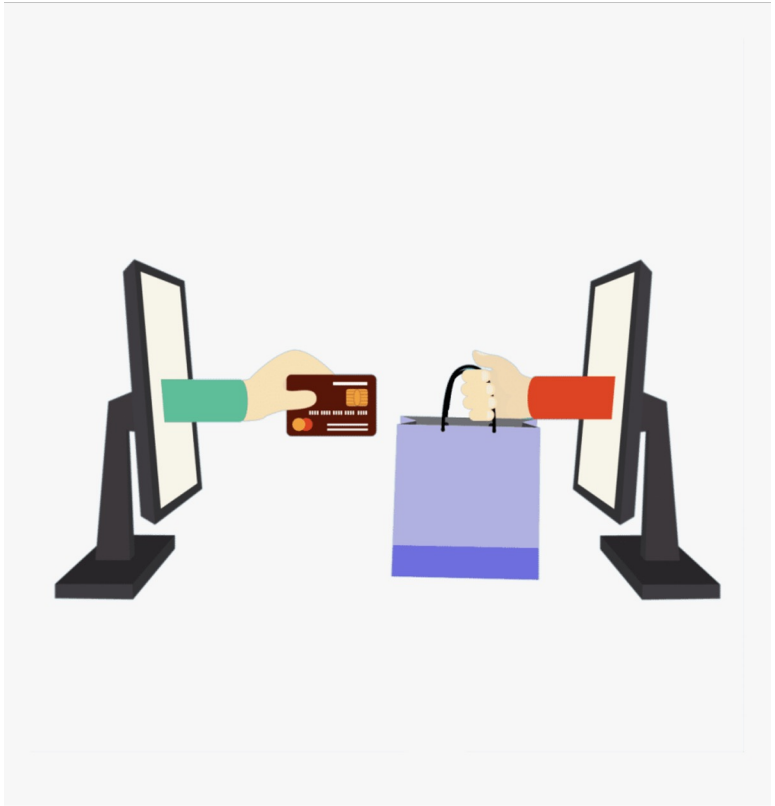


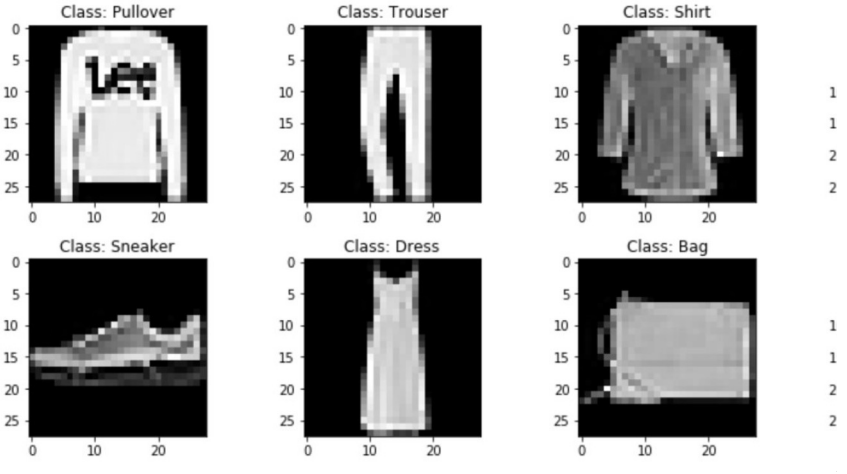
Figure 1

Goal

Image Classification
(Fashion-MNIST Dataset)

Feature importance using Class Activation Mapping (CAM)

Similarities between models?



Related Work

[1] Jia and Colleagues utilized a pre-trained CNN model to classify skin lesions

- Skin lesion image ⇒ CAMs created ⇒ Cropped import regions ⇒ CNN (re-training)
- Increased accuracy of model
- Described CNN and CAM implementation and effectiveness

[2] Agarap and Abien implemented a CNN + SVM model on MNIST and Fashion MNIST data

- MNIST/Fashion-MNIST ⇒ CNN feature vectors ⇒ SVM Classification
- Achieved test accuracy of ~99.04 on MNIST and ~90.72 on Fashion MNIST

CNN Model

Hyperparameters

- Number of epochs: 10
- Batch size: 64
- Learning rate: 1e-3
- Optimizer: Adam (loss=categorical cross-entropy)

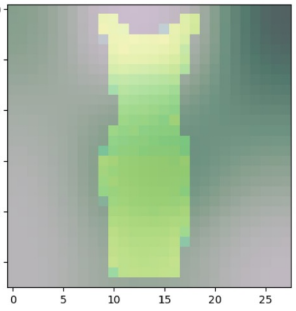
```
#64= num of convolutions defined,size of conv network, relu function
model = Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input_shape= (28,28,1)))
model.add(MaxPooling2D((2,2)))  # 2 by 2 in. shape, 2x2 array of pixels and pick
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64, (3,3),activation='relu'))
model.add(GlobalAveragePooling2D())
#model.add(Flatten()) #flatten input
#model.add(Dense(64, activation='relu')) #64 neurons/equations,
model.add(Dense(10, activation='softmax')) # 10 neurons defined in output layer
```

Model accuracy: 90.86%

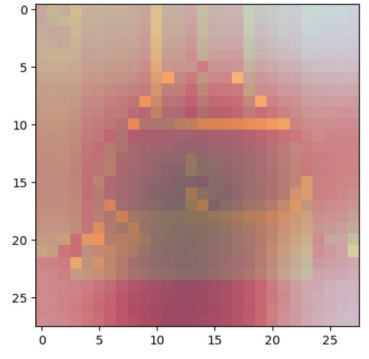
CNN + CAM

```
def show_cam(test_img,
             features_for_img,
             results):
  # get the class with the highest output probability
    prediction = np.argmax(results)
  # get the gap weights at the predicted class
    class_activation_weights = gap_weights[:, prediction]
  # upsample the features to the image's original size (28 x 28)
    class_activation_features = sp.ndimage.zoom(features_for_img, (28/3, 28/3, 1), order=2)
  # compute the intensity of each feature in the CAM
    cam_output  = np.dot(class_activation_features,class_activation_weights)
    print('Predicted Class = ' +str(prediction)+ ', Probability = ' + str(results[prediction]))
    #print('Probability = ' + str(results[prediction]))
  # show the upsampled image
    plt.imshow(np.squeeze(test_img), alpha=0.5)
  # strongly classified (95% probability) images will be in green, else red
    if results[prediction]>0.95:
        cmap_str = 'Greens'
    else:
        cmap_str = 'Reds'
  # overlay the cam output
    plt.imshow(cam_output, cmap=cmap_str, alpha=0.5)
  # display the image
    plt.show()
```

```
[[5.71967466e-05 2.80723384e-06 1.28106080e-07 9.99935031e-01
  2.86808834e-08 3.88904797e-10 4.41703469e-06 1.15610022e-10
  3.55207703e-07 1.29032465e-08]]
Predicted Class = 3, Probability = 0.99993503
```

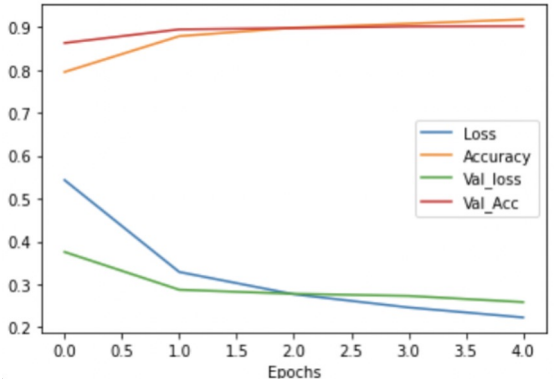


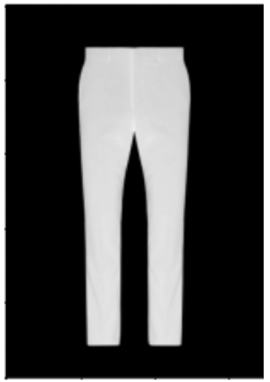
```
[[3.6485912e-03 1.5661485e-02 2.6116975e-02 3.3838924e-03 6.0819829e-04
  6.2228759e-07 6.6652209e-03 2.9943407e-05 9.4206518e-01 1.8199140e-03]]
Predicted Class = 8, Probability = 0.9420652
```



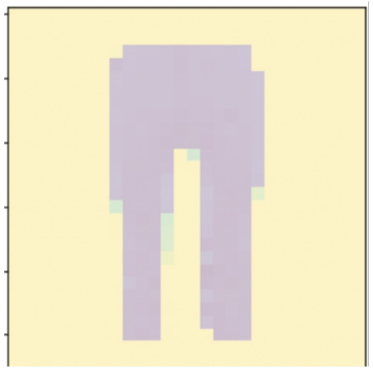
CNN+SVM

Layer (type)	Output Shape	Param #
conv2d_input (InputLayer)	[(None, 28, 28, 1)]	0
conv2d (Conv2D)	(None, 28, 28, 16)	160
max_pooling2d (MaxPooling2D)	(None, 14, 14, 16)	0
conv2d_1 (Conv2D)	(None, 14, 14, 32)	4640
max_pooling2d_1 (MaxPooling2	(None, 7, 7, 32)	0
conv2d_2 (Conv2D)	(None, 7, 7, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 3, 3, 64)	0
conv2d_3 (Conv2D)	(None, 3, 3, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 1, 1, 128)	0
conv2d_4 (Conv2D)	(None, 1, 1, 256)	295168
global_average_pooling2d (Gl	(None, 256)	0
dense (Dense)	(None, 10)	2570





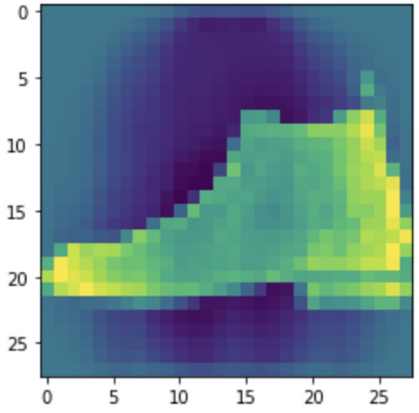
	precision	recall	f1-score	support
0	0.85	0.88	0.86	1000
1	0.99	0.98	0.99	1000
2	0.87	0.87	0.87	1000
3	0.91	0.92	0.92	1000
4	0.85	0.88	0.87	1000
5	0.98	0.98	0.98	1000
6	0.77	0.72	0.74	1000
7	0.96	0.98	0.97	1000
8	0.98	0.98	0.98	1000
9	0.98	0.96	0.97	1000
accuracy			0.92	10000
macro avg	0.91	0.92	0.91	10000
weighted avg	0.91	0.92	0.91	10000



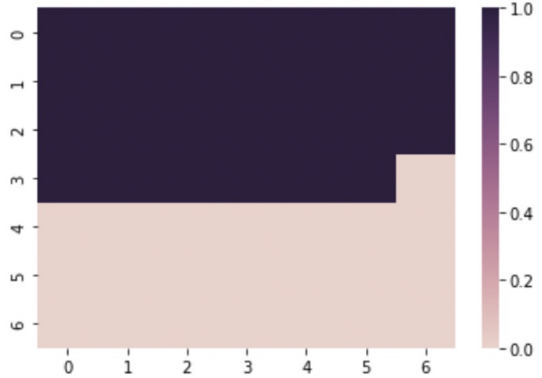
SVM+Mask

	precision	recall	f1-score	support
Ankle Boot	0.94	0.95	0.94	1000
Bag	0.96	0.97	0.96	1000
Coat	0.81	0.85	0.83	1000
Dress	0.87	0.91	0.89	1000
Pullover	0.82	0.79	0.80	1000
Sandal	0.96	0.94	0.95	1000
Shirt	0.71	0.65	0.68	1000
Sneaker	0.91	0.93	0.92	1000
T-shirt/Top	0.82	0.84	0.83	1000
Trouser	0.99	0.97	0.98	1000
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000

'Ankle Boot'

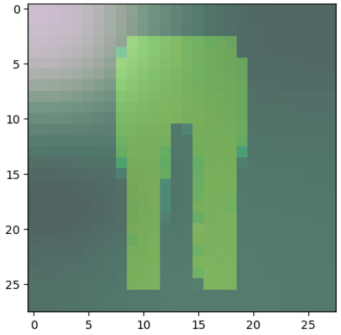


<AxesSubplot:>

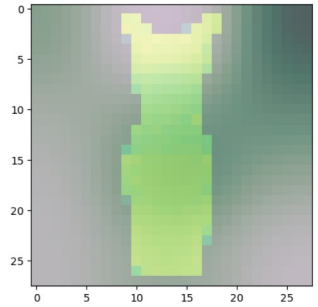


Results

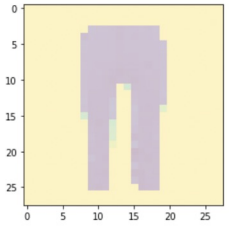
```
[[2.7093931e-06 9.9999690e-01 2.1948605e-07 5.2675297e-10 4.5311719e-09
  1.2960484e-15 2.2344841e-07 8.8702849e-20 1.8335720e-09 2.4847234e-14]]
Predicted Class = 1, Probability = 0.9999969
```



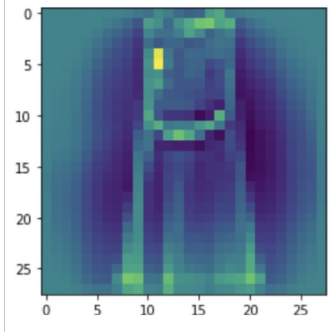
```
[[5.71967466e-05 2.80723384e-06 1.28106080e-07 9.99935031e-01
  2.86808834e-08 3.88904797e-10 4.41703469e-06 1.15610022e-10
  3.55207703e-07 1.29032465e-08]]
Predicted Class = 3, Probability = 0.99993503
```

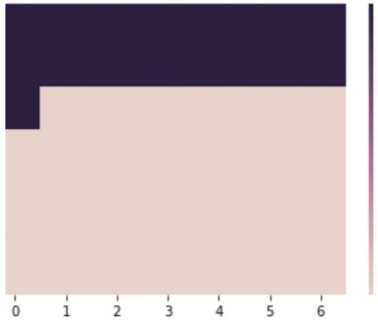


```
[[ 0.16069904  0.2051855   0.13830656  0.08345777  0.11523289  0.0375183
   0.18300316 -0.0068561   0.06824257  0.01521032]]
Predicted Class = 1
(28, 28)
```



'Dress'





Contributions

Anuva

- CNN implementation
- Preprocessing function for google images
- Modification of CAM for google image inputs (according to model)
- SVM+Mask Model
  - Preprocessing
  - SVM classifier implementation

Bhanu

- CNN and SVM implementation
- Modification of CAM for google image inputs (according to model)
- SVM+Mask Model
  - Masking
  - Heatmap generation from prediction

Conclusion

- CAM predicted similar spatial areas for both CNN and CNN+SVM models
  - Identification of patterns in feature importance
    - Allow for recognition of unusual areas ⇒ debugging

Future Work

- Class-wise analysis of accuracy
- Deployment of model with obstructed images
  - Create own dataset as input for model

References

Images:

Figure 1: https://www.kindpng.com/picc/m/121-1219048_online-shopping-clipart-png-transparent-png-png-download.png

[1] Jia, X., and Shen, L. (2017, March). *Skin Lesion Classification Using Class Activation Map*. Retrieved from https://arxiv.org/ftp/arxiv/papers/1703/1703.01053.pdf

[2] Agarap and Abien. (2017). *An architecture combining convolutional neural network (CNN) and support vector machine (SVM) for image classification*.Retrieved from https://arxiv.org/pdf/1712.03541.pdf