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

Anuva Suresh
Michigan State University
sureshan@msu.edu

Bhanu Kanarmarlapudi
Michigan State University
kanamar1@msu.edu

***Abstract*—Deep learning is a technique utilized in a wide range of fields. The most applicable and useful class of deep neural networks for image processing, however, is a Convolutional Neural Network (CNN). One of the most interesting industries where CNNs are heavily used is fashion. CNN's are used in the fashion industry to aid in e-commerce through clothing recommendations and recognition. To understand how a neural network predicts the class of any particular clothing item, it is important to visually analyze the pixels/areas in an input image providing the most contribution to the model's prediction. One such method used to perform this is Class Activation Mapping (CAM). This study implements CAM with a CNN model and a CNN+Support Vector Machine (SVM) hybrid model to decipher whether the CAM outputs show similarities. Additional to these two models, a traditional SVM model is attempted by masking the input image with a custom mask and visualizing the model's accuracy using a heatmap by moving the mask all over the image. In visualizing areas of importance, it will be possible to show that, regardless of the image classification method, particular spatial features of a clothing item contribute to its likelihood of belonging to a specific class. This, in turn, allows researchers to identify whether a new classification model is focusing on unimportant areas and debug the model's decision process.**

I. Introduction

The fashion industry is constantly changing at a rapid pace in how consumers are able to shop or view clothing items. Image classification is mainly used in this industry to help buyers and sellers in e-commerce. A prime example of an event in which image classification would be beneficial is when sellers upload images of items they would like to buy. The seller must apply tags (keywords) that describe the fashion item so that consumers can easily locate what has been uploaded. However, far too often, sellers mislabel items or provide inaccurate keywords. To prevent this, image classification methods can provide the seller with accurate tags which truly describe the item being uploaded. Now, when consumers search for items that contain those tags, they will be directed to the seller's uploaded image.

Convolutional Neural Networks (CNNs) are fully connected feed-forward neural networks that have been shown to be most effective in performing image classification due to their multiple layers including convolutional layers, which reduce the dimensionality of input images without the loss of any information. Images are known to have a very large number of parameters, so this capability of the CNN is extremely beneficial. In standard deep neural networks, if there are a large number of layers and neurons which are connected to one another, it means that there will be a large number of parameters that need to be learned. This increases complexity and computation which may lead to overfitting. For image data, there may be many pixels and features in the input image which do not really contribute to the actual labeling of the main object in the image. This may be pixels that are for the background or miscellaneous items in the image. Standard deep neural networks take all of this into account which further increases complexity. CNNs, however, make use of convolution operations, like filtration and max pooling to decrease data dimensionality and focus on the most important features of the image which leads to a dramatic decrease in computational and network complexity.

Feature importance is another important aspect of image classification. When a model is implemented to complete image classification, the outcome, and accuracy of this model in labeling the input image depend on the areas that are most important. Which areas of the image hold the heaviest weight and contribution to which label? If this is known, then those who have worked to implement this model can see why accuracy is exceptional or poor. They will be able to decipher whether unusual areas are given more importance and debug this decision process in the neural network. This study focuses on the outcome of applying CNN and a hybrid model of CNN with a traditional machine learning model known as the Support Vector Machine (SVM) on the Fashion MNIST Dataset, a dataset including a large number of fashion items available for image classification. These models are applied in conjunction with Class Activation Mapping (CAM) to visualize which areas of an input image are truly the most important to each model. A third approach to identify the important areas was also attempted by using an SVM classifier and masking the input image with a small mask that runs over the entire image. The output from calculating the accuracies of the SVM at each step can be visualized with a heatmap. Test images are applied to each CNN model with CAM visualizations to observe whether or not the same features are given the highest importance across all implementations. As mentioned above, this allows researchers to see what features are truly important and recognize when unusual features are given significant importance.

This paper will explain CAM, present work related to class activation visualization and other methods, describe the Fashion MNIST Dataset, present the framework of the models implemented in this study, describe the results obtained from the implementations (including a discussion of possible reasons), present a conclusion, describe the individual contributions made to this study, and present possible future work.

II. Class Activation Mapping (CAM)

Class Activation Mapping (CAM) is a technique used in the field of computer vision to visualize the importance of features for an image classified by a CNN. Saliency maps can provide information on which parts of an image are most important to a CNN, but CAMs are class-discriminative saliency maps. This means CAMs distinguish between classes (labels) and provide information specific to each class [4]. In other words, CAMs allow one to visualize, for each class, which features the CNN applies most consideration when assigning that particular class to the input image.

CAM utilizes a global average pooling layer (GAP) immediately after the last convolutional layer and before the final classification layer. The activation map produced by the CAM is created using weights from the final fully-connected layer of the CNN. Global average pooling is applied to each feature map (the average is computed for every pixel in the feature map). A vector of scalars is created with these averages. The final layer uses the output of the GAP to learn a linear model with corresponding weights for each of the classes. The weights for each of the classes are used to weight each feature map and create class activation maps. The various weighted combinations applied to the same feature maps give rise to class activation maps for each of the specific classes [1].

III. Related Work

Reference [2] focuses on utilizing a pre-trained CNN model to classify skin lesions and CAM to further increase classification accuracy. Similar to our method, CAM is applied to the CNN after an image is sent as input. Once the CAMs for the image is produced, the import regions (heavily important) regions are cropped and once again sent back into the CNN for retraining. While this paper did not follow this method exactly, the process of applying the CNN and CAM as well as taking into consideration the importance of features was the same. Additionally, the equation considered for CAM was also considered in this study.

$$M_c = \sum_k w_k^c f_k(x, y)$$

Where c is the class, M_c is the CAM of class c, $f_k(x, y)$ is the activation value of unit k in the last convolutional layer at (x,y) spatial location, and w indicates the weight importance of f_k for specific class c.

Reference [3] focuses on implementing a CNN for image classification. The paper mainly describes the working algorithm of CNN on image classification and reports accuracies as well as MSE. The structure of CNN implemented in this study is not the same as what is implemented in this reference in that this reference utilizes a different number of layers, but the underlying structure of the CNN implementation is the same.

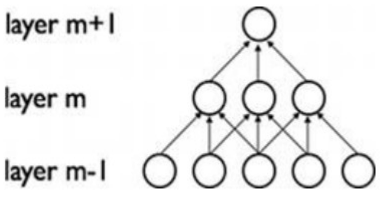


Fig. 1. Feed-forward network of convolution and sub-sampling layers to classify images

Fig.1 represents a feed-forward network of convolution and sub-sampling layers to classify images. [8] talks about the combination of CNN with SVM for classification. The paper mainly discusses the architecture of the models and their performances. The idea of implementing a hybrid model was inspired by this paper but our model was implemented differently from the paper.

The concept of a Class Activation Map was introduced by Zhou et al in the paper Learning Deep Features for Discriminative Localization [9]. They have defined the term Class Activation Maps to refer to weighted activation maps generated by CNN models. These weighted activation maps lead to the prediction of a specific label for the image.

IV. Data Set

The data set used for this project is Fashion MNIST [6], which is an open and easily accessible dataset. The dataset consists of 60,000 samples for training and 10,000 samples for testing. It has samples belonging to 10 different classes namely T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. Fig.2 represents a small set of samples present in the dataset.

Each image in the dataset is of size 28×28. The dataset is perfectly balanced having the same number of samples for all the classes in both testing and training datasets.

V. Models

All models were implemented using Python packages. The following subsections are the model descriptions of the implemented models.

A. CNN model

The CNN architecture in this study was implemented using Tensorflow (a machine learning library) and Keras API which includes the modules necessary to build the CNN. Hyperparameters of the model are as follows:

- Number of epochs = 10

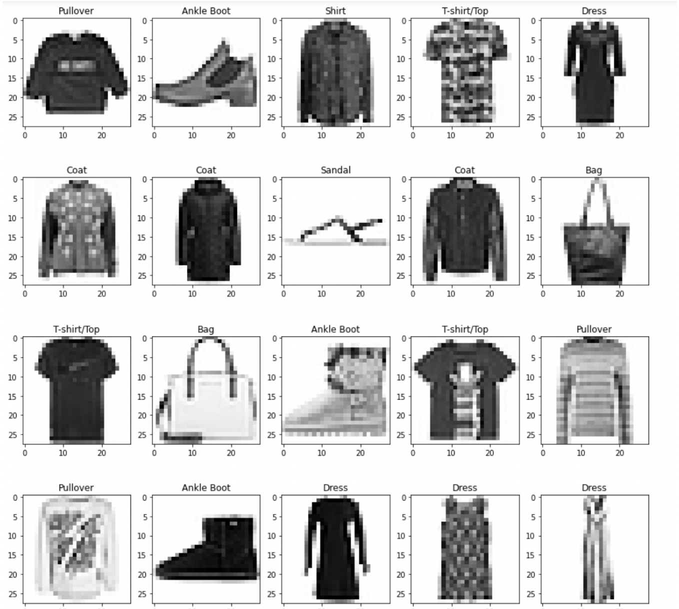


Fig. 2. Fashion MNIST

- Batch size = 64
- Learning rate = 1e-3
- Optimizer = Adam (loss = categorical cross-entropy) as labels are one-hot encoded

The architecture of model:

```
model.Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input shape=(28,28,1)))
model.add(MaxPooling2D((2,2)))
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(Dense(10, activation='softmax'))
```

Model accuracy = 90.86%

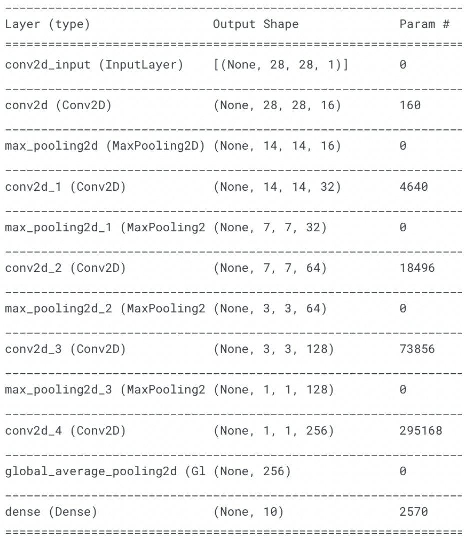
A preprocessing function was also included for google images (images that were not used in training or testing sets for models). This allowed for the use of the obtained dress and trouser images on CNN and CAM. Images were sent to CNN to see if the correct classification was produced.

For the CAM implementation:

- Created CAM model using Model from Keras taking into account the final convolution layer and dense layer (activation=softmax) outputs
- Weights were obtained from the dense layer
- Features for the input image were obtained at index 0
- Features were scaled to original image size $h \times w$
- Weights were initialized which are used for a specific class (0...9)
- The dot product was calculated between the scaled image features and the weights for one class to obtain the input image array
- show_CAM function was implemented from an outside source (included in a notebook) to show the probability of an object belonging to the class and its activation map
- The input image is preprocessed before sending to show_CAM function
- An Activation visualization is created (darker areas indicating more importance).

B. CNN+SVM model

In the CNN+SVM architecture, the feature vectors generated by the CNN are fed as inputs to the SVM for classification. It is implemented using Tensorflow (a machine learning library), sklearn, and Keras API which includes the modules necessary to build the CNN and SVM models. Fig.3 represents the summary of the CNN model. Hyperparameters of the CNN model are as follows:



```
_________________________________________________________________
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
=================================================================
```

Fig. 3. CNN Model Summary

- Number of epochs = 5
- Batch size = 32
- Learning rate = 1e-3
- Optimizer = Adam (loss = categorical cross-entropy)
- loss=sparse_categorical_crossentropy

The hyperparameters of the SVM model are:

- C=10
- kernel=rbf
- gamma=auto
- probability = True

The images collected from Google were used as test images. This model uses the previously described preprocessing image function to normalize the image. The normalized image is sent to CNN to compute the feature vectors. These vectors are then used to identify their class using the SVM model. The cam model is developed by making use of the last convolution layer. The features and predictions are used to generate the class activation mappings. The previously defined show_CAM function is used to identify the spatial areas of the image.

C. SVM+Mask model

The third model we have implemented is a traditional SVM model and computing the performance of the model by applying a mask to small portions of image. The data was normalized by dividing and computing the mean. The mean was then subtracted from the data. The Hyperparameters used for the SVM classifier are: The hyperparameters of the SVM model are:

- C=10
- kernel=rbf
- gamma=auto
- probability = True

A 4×4 mask was created by initializing all zeros. This mask when then applied to the image, starting at the top left to bottom right by moving right and down. The masked image at each step was used as a test image to make a prediction. The final predicted output size for the input of 28×28 is 7×7. A heatmap is visualized from these predictions to identify the key feature areas and the most contributing feature locations for the classifier to make the decision.

VI. Results

The following are the results and visualizations generated by the above-discussed classification models.

A. CNN model

The performance of the CNN model with above-stated hyperparameters after 10 epochs is as follows:

- Loss: .1643
- accuracy: .9385
- validation loss: .2742
- validation accuracy: .9086 (90.86%)

Dress, Trouser, and Bag images which are collected from Google are fed as inputs to CNN. It's observed that all three of them were classified correctly. Fig.4 and Fig.5 represent the class activation mappings for the Trouser and Dress images respectively.

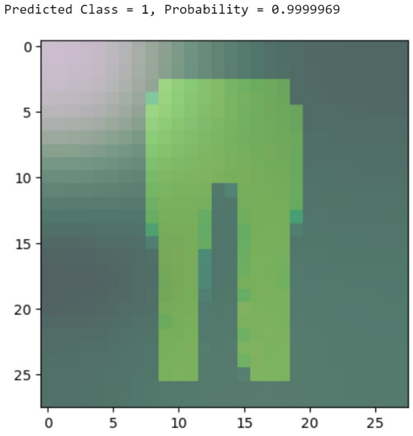


Fig. 4. Activation visualization for googled trouser image.

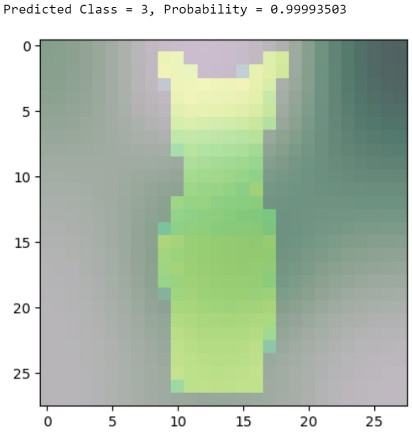


Fig. 5. Activation visualization for googled dress image.

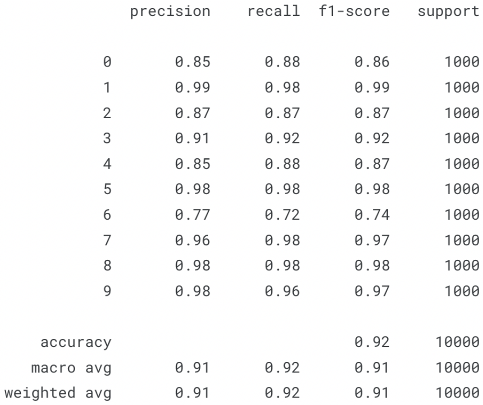
As shown, the probability of each item belonging to their class is 99%. The overall shape and all areas of the trouser image are taken into equal consideration for CNN to place this item correctly in class 1. This is due to the fact that there is no design specificity of the trouser and the overall shape of the trousers are taken into consideration. The dress was also labeled correctly in class 3. For the dress, the waistline is given the most importance (darkest shading). This is due to the fact that dresses tend to have a more curved feature in the waistline. CNN must have learned the presence of curves in dress images and this is why the area is given so much importance.

B. CNN+SVM model

The CNN model performance after 5 epochs is

- loss: 0.2169
- accuracy: 0.9195
- val_loss: 0.2512
- val_accuracy: 0.9068

The feature vectors generated by CNN are then utilized as inputs to the SVM model for training and prediction. Fig.6 represents the classification report of the SVM classifier.



```
              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
```

Fig. 6. Classification Report of SVM

It can be seen from the above classification report that the accuracy of the model prediction increased slightly. Fig.7 is the final CAM visualization generated after preprocessing the input trouser image and gathering feature vectors, making use of them to predict their category and visualizing using the show_CAM function.

```
Predicted Class = 1
(28, 28)
```

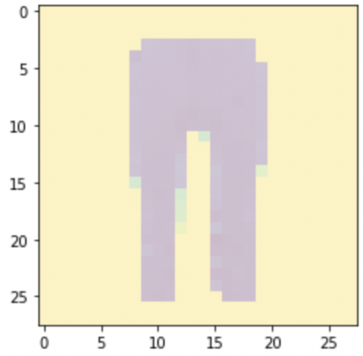
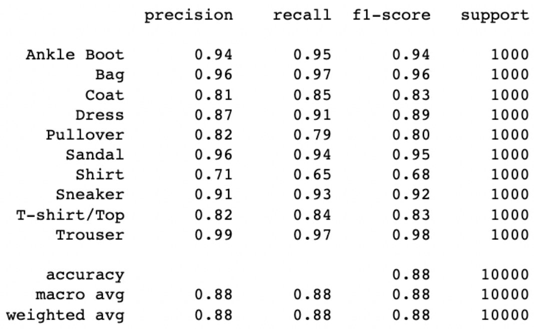


Fig. 7. Activation visualization for Googled Trouser Image

C. SVM+Mask model

The data was initially normalized and the target categories were labeled. The normalized data was sent as input to the SVM classifier and Fig.8 shows the classification report of the model prediction on the test data.



```
              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
```

Fig. 8. Classification Report of SVM+mask Model

Fig.9 and Fig.10 are the heatmaps for a randomly selected Ankle Boot and Dress from the test data. The darker shade of cells of the heatmap denotes correct prediction. The lighter shade of cells denotes incorrect prediction. It can be inferred from Fig.9 that the model predicted accurately when the background cells where the pixels of the Ankle Boot are not present are masked and inaccurately when the pixels of the Ankle Boot are present. However, that is not the case with Fig.10.

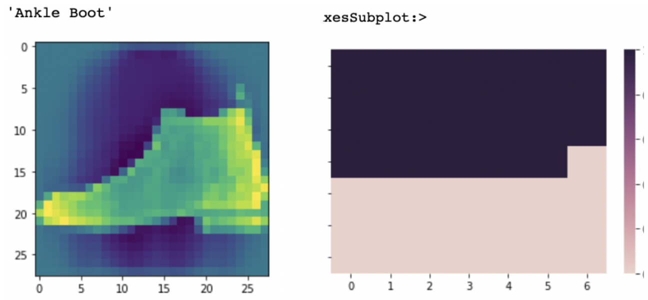


Fig. 9. Ankle Boot and it's heatmap

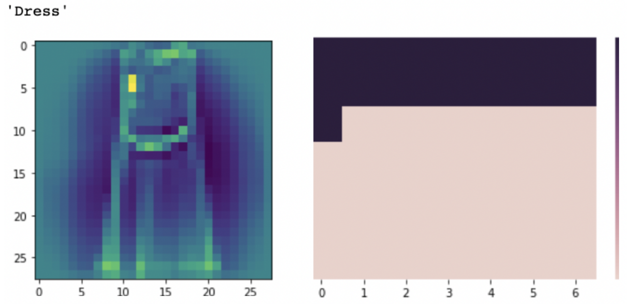


Fig. 10. Dress and it's heatmap

VII. Conclusion

The class activation mapping (CAM) technique identifies the key spatial areas well for both convolution models. CNN+SVM model has slightly better accuracy compared to the CNN model. Both models have predicted the Trouser image collected from Google correctly and also generated similar activation visualizations. The variations between the heatmaps generated by the CAM models are due to the implementation and architectural differences between the CNN and CNN+SVM models. Even though the SVM+Mask model generates a simple heatmap, the similarities between the CAM and the heatmap generated by the SVM+Mask model for the dress image are quite similar. The darker shade on the CAM output represents the importance of the features. It can be observed that the waist area has more important spatial features in the CAM output generated for the dress. The heatmap technique doesn't identify spatial areas as well compared to class activation maps, but this method can still be used to generalize and identify key areas of the image that contribute to correct or incorrect classifications. The people working on the image classification tasks day-to-day can utilize the above-implemented CAM and heatmap techniques to easily identify the patterns that are not contributing much to classification tasks.

VIII. Contributions

We are a team of two, and we have tried to split the work equally among ourselves. Following are the individual contributions. Both of us equally contributed to prepare the final report and presentation.

A. Anuva

- CNN Model Implementation
- Preprocessing function for Images collected from Google
- Modification of CAM for google image inputs (according to model)
- SVM+Mask Model
  - Preprocessing
  - SVM Classifier Implementation

B. Bhanu

- CNN + SVM Model Implementation
- Modification of CAM for google image inputs (according to model)
- SVM+Mask Model
  - Masking the image
  - Heatmap generation based on prediction

IX. Future Work

In the future, it will be interesting to perform a class-wise analysis of accuracy. In other words, the detection of which fashion items are hardest to classify and an investigation as to why. Additionally, another experiment could be carried out to observe the outcome of the model when it is deployed in the wild without having backgrounds distinctive to that which is present in the Fashion MNIST Dataset. What would happen if the images did not only have a plain background and there were other items in the background of images. How can this model trained on the Fashion MNIST Dataset be refined to classify these images accurately? An original dataset could be created with obstructed images to train this model. Regardless, this study certainly raises many image classification questions which should continue to be researched.

References

[1] H. Jung and Y. Oh, "Towards Better Explanations of Class Activation Mapping," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), October 2021.

[2] X. Jia and L. Shen, "Skin Lesion Classification using Class Activation Map", March 2017.

[3] D. Jaswal, S. V, and K. P Soman, "Image Classification Using Convolutional Neural Networks," International Journal of Scientific and Engineering Research, vol. 5, Issue 6, June 2014.

[4] Bae, Wonho, Junhyug Noh, and Gunhee Kim,"Rethinking class activation mapping for weakly supervised object localization," European Conference on Computer Vision. Springer, Cham, 2020.

[5] Greeshma, K V and K. G. Sreekumar, "Fashion-MNIST Classification Based on HOG Feature Descriptor Using SVM" (2019).

[6] Xiao, Han, Kashif Rasul, and Roland Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv preprint arXiv:1708.07747 (2017).

[7] Meshkini, Khatereh, Jan Platos, and Hassan Ghassemain, "An analysis of convolutional neural network for fashion images classification (Fashion-MNIST)," International Conference on Intelligent Information Technologies for Industry, Springer, Cham, 2019.

[8] Agarap, Abien Fred, "An architecture combining convolutional neural network (CNN) and support vector machine (SVM) for image classification," arXiv preprint arXiv:1712.03541 (2017).

[9] Zhou, Bolei, et al, "Learning deep features for discriminative localization," Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.