CSE 674 : Advanced Machine Learning

PROJECT - 3
Explanation based Writer Verification

❖ Abstract

We present http://betelgeuse.cse.buffalo.edu/and_dataset (XAI-AND) a dataset that contains 13700 handwritten samples and 15 corresponding expert examined features for each sample. Furthermore, we propose Deep Learning and explanation based systems that can assist human examiners to detect authentic or forged handwritten samples. We describe our approach based on the word "AND" extracted from full-page handwritten samples in CEDAR Letter data. Our first approach is based on learning the mapping between expert features and the input images using the soft probabilities learned by a Features Learner Network (FLN). In the second, we use a Probabilistic Graphical Model (PGM) in tandem with FLN to generate log-likelihood of samples coming from the same or different source. We perform various experiments based on variations of our two approaches and evaluate the results based on Type 1 Accuracy (T1), Type 2 Accuracy (T2) and Likelihood Ratio (LR). The dataset is released for public use and the methods can be extended to provide explanations on other verification tasks like face verification and biomedical comparison. The code is available on https://github.com/mshaikh2/HDL_Forensics.

Another approach that we propose is the visual explanation of the decisions made by Convolutional Neural Networks by using a visualization technique called Gradient-weighted Class Activation Mapping (Grad-CAM). They provide a high-resolution class-discriminative visualization. In the context our classification models, these visualizations would (a) lend insights into failure modes of these models (what factors led the model believe that the two image fragments came from the same/different writer), (b) are more faithful to the underlying model, and (c) help achieve model generalization by identifying dataset bias.

Finally, we propose to design and conduct human studies to measure if Grad-CAM explanations help users establish appropriate trust in predictions from deep networks and show that Grad-CAM helps untrained users successfully discern a 'stronger' deep network from a 'weaker' one even when both make identical predictions.

❖ Introduction

The goal of Handwriting verification is to find a score of similarity between the two given handwritten samples. Researchers have successfully approached the problem of identifying inter-writer and intra-writer variations to solve the task of verification. However, these approaches were based on conventional machine learning and purely

handcrafted features. The forensic community has used the Likelihood Ratio (LR) to denote the degree of similarity between a Known k and Questioned q sample. With the advent of Deep Learning, where the features are automatically learned, the solutions of this problem became a more black box. Siamese Networks and Triplet Loss have attempted to solve the problem of isolating similar and dissimilar samples. However, the predictions of such systems raise eyebrows in the expert community, as they do not have a clue why those predictions were made. When a human examiner compares forgery with real, he does it based on tangible features based on which a judge can provide quality results. If the expert examiner merely states yes or no then his integrity will be questioned. This instance applies to even an intelligent assistant of the human examiner. Our proposed system provides such an intelligent assistant in the form of Explainable Artificial Intelligence interface (XAI).

Many organizations like DARPA have invested heavily in providing explainable based systems. Researches have been working on why a classifier's output should be trusted and developed techniques like LIME to explain the predictions of any classifier in an interpretable and faithful manner. There also have been attempts to trace the predictions of the model back to the training data for understanding model behavior, debugging models, detecting dataset errors, and even creating visually indistinguishable training-set attacks. With these advances, the need for an interface that can provide trustworthy predictions is imminent in the forensic community.

Many communities including Healthcare and Biomedical cannot completely rely on black box predictions and require more explanation as to why a decision was made by the AI. These communities need more understanding about the predictions of the black-box so that they can assimilate their current surrounding context and carefully utilize the intelligent explanations of the system and make an informed decision. The proposed system learns from human observed annotations and takes one step towards helping the forensic document examiner (FDE) make a rational examination.

Although the system exhibits extremely good performance, it fails to explain the predictions of convolutional neural networks, as they lack the interpretability, that is, they are unable to explain which features or which part of the image contributed in making the final decision by the model. In order to build trust in intelligent systems and move towards their meaningful integration into our everyday lives, it is clear that we must build ‘transparent’ models that explain why they predict what they predict. It would, therefore, be helpful to be able to explain why a model made the prediction it made. For example, when a model predict that the two images came from the same writer when, in reality, the two images are written by different writers, it is hard to say why the wrong judgement was made without visualizing the network’s decision. Visualizations can also build confidence about the predictions of a model. For example, even if a model correctly predicts that the two images were written by different writers, we would want to confirm

that the model bases its decision only on the distinguishing features of the “AND” image and not some other object or background pixels.
Convolutional Neural Networks (CNNs) and other deep networks have enabled unprecedented breakthroughs in a variety of computer vision tasks, from image classification to object detection, semantic segmentation, image captioning, and more recently, visual question answering. A visualization technique called Class Activation Map(CAM) visualization can be deployed for this purpose. We intend to identify the pixel locations in the image of handwritten word “and” that correspond to a certain feature (for instance, for the feature “staff of a”, we need to identify the pixel locations that contain staff of ‘a’ as seen by the model.
We also aspire to implement this technique for the writer verification task to visualize which features in the image of handwritten word “and” were responsible for the decision (whether the pair of images are written by same writer or not) made by the model.

❖ DATASET

XAI-AND dataset is a publicly available dataset for handwriting verification, comprising of 15,518 “AND” image fragments extracted from CEDAR Letter Dataset written by 1567 writers. Each “AND” image fragment is labeled by a questioned document (QD) examiner with 15 explainable discrete features. QD examiners have specified these handwriting features based on years of training using seven fundamental elements of handwriting as shown below. We have created a web based truthing tool for QD examiners to enter the values for the 15 features for “AND” images fragments. The data entry work using the truthing tool was shared primarily between 89 external examiners. The data entered by the external examiners was verified by 2 QD examiners. The resultant dataset serves as a good resource for explanation based handwriting verification.

Values → Features	**1**	**2**	**3**	**4**
Pen Pressure	Strong	Medium		
Letter Spacing	Less	Medium	High	
Size	Small	Medium	Large	
Dimension	Low	Medium	High	
Is LowerCase	No	Yes		

Is Continuous	No	Yes		
Slantness	Normal	Slight Right	Very Right	Left
Tilt	Normal	Tilted		
Entry stroke "A"	No Stroke	Downstroke		
Staff of "A"	No Staff	Retraced	Loopy	Tented
Formation "N"	No Formation	Normal		
Staff of "D"	No Staff	Retraced	Loopy	
Exit stroke "D"	No Stroke	Downstroke	Curved Up	Straight Across
Word formation	Not Well Formed	Well Formed		
Constancy	Irregular	Regular		

❖ METHODS

1. Feature Based Explanations

■ Feature learning network (FLN)

We introduce a learning network such that Encoding 5 is processed in FLN and its output is then supplied to Decoding 4 for reconstruction. FLN consists of learning units for each of the 15 categories present in the dataset. Each learning unit comprises of a hidden layer (H) and a SoftMax layer (SM) such that there is one H and SM for each task. Each H has 128 neurons and each SM has neurons corresponding to the number of classes in respective task. The sum of Categorical-Cross Entropy (CCE) loss for each SM is backpropagated during the training. Hence the loss of FLN is denoted by:

$$L_{FLN} := \sum_{j} - \sum_{i} (y_i' \log(y_i) + (1 - y_i') \log(1 - y_i))$$

where *i* is the class is each task category denoted in Table 2, 3 and 4. and j ranges from 1 to 15 for each of the categories. Next, all the H's are concatenated and input to a fully connected (FC) layer consisting of 512 neurons. The output of FC is then reshaped to 1x1x512 and is then input to Decoding 4 layer.
Hence, instead of passing the Encoding 5 directly to Decoding 4, which normally is the case, we do the following:
(i) fan out Encoding 5 into 15 clone branches,
(ii) learn mapping of each category with the image, as Encoding 5 is a latent representation of the image,
(iii) combine the learned representations of each category to further reconstruct the input image.

Total Loss: Total Loss of the deep learning network is the sum of Reconstruction Loss (L_{Recon}) and FLN Loss (L_{FLN}). Where L_{Recon} is calculated by measuring the models ability to reconstruct the image close to original. To generate the image the Decoding 5 neurons are activated by Sigmoid function since the input was normalized to be between 0 and 1.

$$L_{Recon} := -\sum_{i=1}^{m} y_i ln(p_i) + (1 - y_i) log(1 - p_i)$$

$$L_{Total} := L_{Recon} + L_{FLN}$$

- **Inference models:** Analysis of handwriting features helps a Forensic Document Examiner (FDE) to find the probability (p) that the known (k_f) and questioned (q_f) handwritten samples were written by the same writer. Each handwritten sample, q_f and k_f is associated with 15 discrete features ***f*** = $\{f_1, f_2, \ldots f_{15}\}$. We have used two methods for analysis of handwritten features:
  1. **Distance as a measure (DAAM):**
     We use Cosine Similarity (C_{sim}) as measure the degree of similarity. We measure the similarity of the soft classes learned by the FLN softmax layer by computing the cosine similarity between the corresponding categories of two input images.

$$C_{sim}(q_{f_j}, k_{f_j}) = \frac{\sum_{i=1}^{n} q_{f_{j_i}} k_{f_{j_i}}}{\sqrt{\sum_{i=1}^{n} q_{f_{j_i}}^2} \sqrt{\sum_{i=1}^{n} k_{f_{j_i}}^2}}$$

2. **Likelihood as a measure (LAAM):**

   Likelihood ratio (LR) is the ratio of the joint probability $P(q_f, k_f | l_0)$ of q_f and k_f given the handwritten samples were written by the same writer l_0 to the joint probability $P(q_f, k_f | l_1)$ of q_f and k_f given the samples were written by different writers l_1. This is computationally expensive and infeasible. Hence, we simplify the calculation by calculating distance between q_f and k_f.

$$C_{OCS}(q_f, k_f) = \sum_{j=1}^{15} C_{sim}(q_{f_j}, k_{f_j})$$

2. Visual Explanations

- It is often helpful to be able to explain why a model made the prediction it made.
- For example, when a model mis-classifies an image, without visualizing the network's decision, it is hard to say why the misclassification was made.
- Visualizations give confidence about the predictions of a model.
- For example, even if a model correctly predicts a car as a car, we would want to confirm that the model bases its decision on the features of a car and not on the features of some other object that might occur together with cars in the dataset (like road, traffic lights, etc).

CLASS ACTIVATION MAP (CAM)

- A technique used for CNN explanation is Visual Explanation. Class Activation Map(CAM) can be deployed for this purpose. CAM is used to identify the discriminative pixel locations by which CNN is able to predict the class.
- For our use case, we intend to identify regions in the image of handwritten word “AND” that correspond to a certain feature (for instance, for the feature “staff of a”, we need to identify the pixel locations that contain staff of ‘a’ as seen by the model).
- We use this technique for the writer verification task to visualize which features in the image of handwritten word “AND” were responsible for the

decision (whether the pair of images are written by same writer or not) made by the model.

- One of the ways of using CAM is by producing heatmaps of class activations over input images. This means that the heatmap will give us the highlighted portion of the image that contains the object of interest (the class we intend to visualize : cat, dog, etc.) A class activation heatmap is a 2D grid of scores associated with a particular output class (say bird, car, road,etc.) computed for every location for an input image, indicating how important each location is with respect to that output class.
- In our case, the output classes could be the 'staff of a', 'staff of d', etc.
- This technique, however, can only be employed for identifying discriminative regions used by a restricted class of image classification CNNs which do not contain any fully-connected layers.
- CAM produces a localization map for an image classification CNN with a specific kind of architecture where global average pooled convolutional feature maps are fed directly into softmax.
- Specifically, let K be the number of feature maps produced by the last layer of the CNN,

  $A^k \in \mathbb{R}^{u \times v}$

- These feature maps are then spatially pooled using Global Average Pooling (GAP) and linearly transformed to produce a score S^c for each class c,

$$S^c = \sum_k \underbrace{w_k^c}_{\text{class feature weights}} \overbrace{\frac{1}{Z} \sum_i \sum_j}^{\text{global average pooling}} \underbrace{A_{ij}^k}_{\text{feature map}}$$

- To produce the localization map for modified image classification architectures, such as above, the order of summations can be interchanged to obtain

$$S^c = \frac{1}{Z} \sum_i \sum_j \underbrace{\sum_k w_k^c A_{ij}^k}_{L_{\text{CAM}}^c}$$

Grad-CAM

- Grad-CAM (Gradient - Class Activation Map) is a specialized technique of achieving the Class Activation Map.

- It is a class-discriminative localization technique that can generate visual explanations from any CNN-based network without requiring architectural changes or re-training.
- Convolutional features naturally retain spatial information which is lost in fully-connected layers, so we can expect the last convolutional layers to have the best compromise between high-level semantics and detailed spatial information.
- The neurons in these layers look for semantic class-specific information in the image (different objects).
- In order to obtain the class discriminative localization map Grad-CAM $L^{c}_{Grad\text{-}CAM} \in R_{u \times v}$ of width u and height v for any class c , we first compute the gradient of the score for class c, y^{c} (before the softmax), with respect to feature maps A^{k} of a convolutional layer, i.e. $\partial y^{c} / \partial A^{k}$.
- These gradients flowing back are global average-pooled to obtain the neuron importance weights α^{c}_{k} :

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

- This weight α^{c}_{k} represents a partial linearization of the deep network downstream from A, and captures the 'importance' of feature map k for a target class c.
- It uses the gradient of target, flowing through the final convolution layer to produce a coarse localization layer highlighting the important regions in the image.
- We apply a ReLU to the linear combination of maps because we are only interested in the features that have a positive influence on the class of interest, i.e. pixels whose intensity should be increased in order to increase y^{c}.

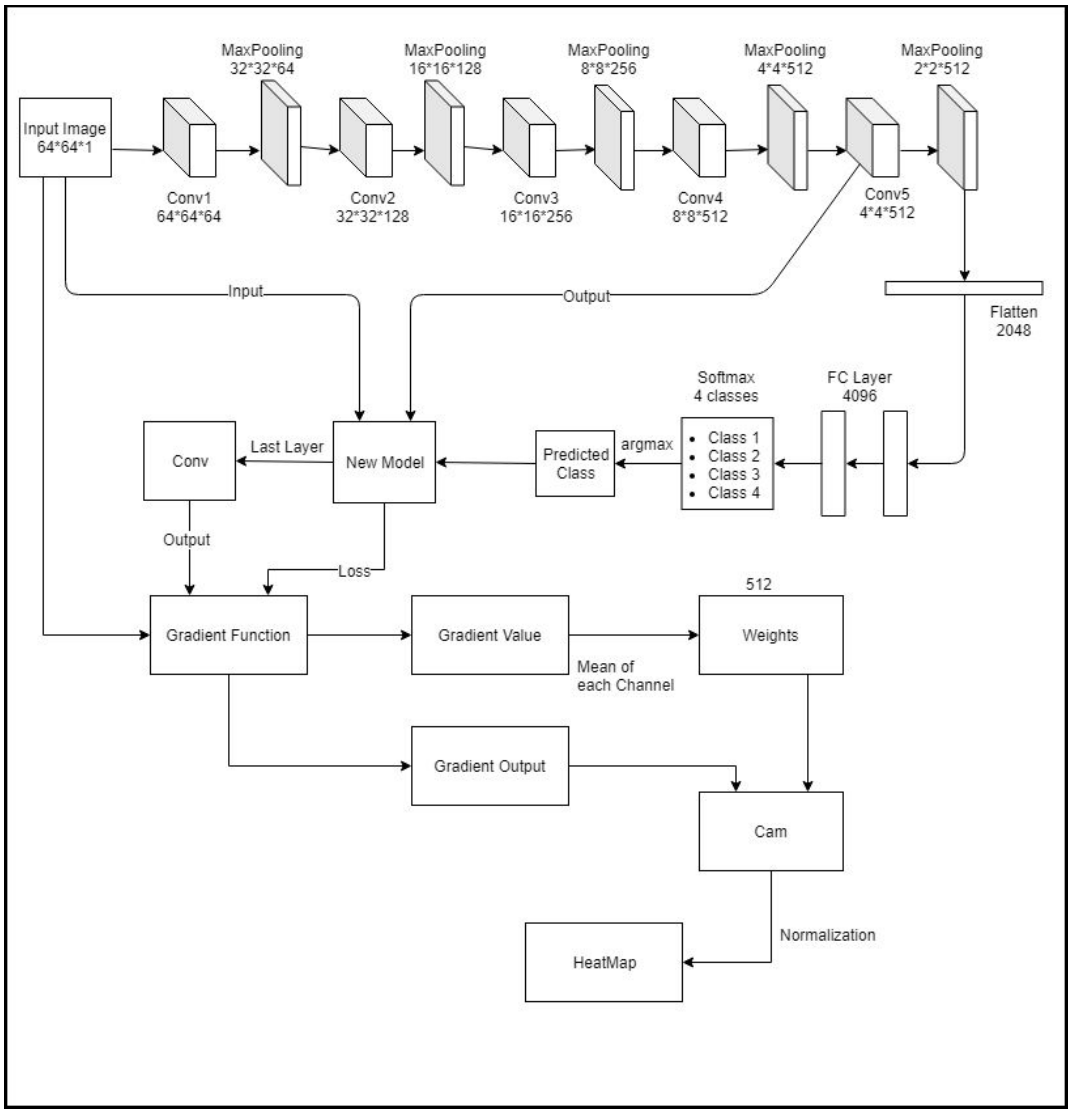
$$L^c_{\text{Grad-CAM}} = ReLU \underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

- Negative pixels are most likely to belong to other classes in the image. Hence, without this ReLU, localization maps sometimes highlight more than just the desired class and achieve lower localization performance.

- CAM has certain restrictions regarding the type of models that it can be used with. But, Grad-CAM overcomes this drawback and can be used, without any architectural changes or re-training, with a wide variety of CNN model families such as:
    - CNNs with fully connected layers
    - CNNs used for structural outputs (e.g. captioning)
    - CNNs used in tasks with multi-model input (e.g. VQA)
    - Reinforcement learning

❖ ARCHITECTURE (Proposed)

Following is the architecture that we have proposed for the handwriting verification task:



1. Input Image

- The input images will be a collection of images of the handwritten word “and” written by multiple writers.
- This would be from the CEDAR dataset of the handwritten word “and” images from over 200 writers.
- The input images will be pre-processed (perform affine transformations like translation, scaling and shifting) in order to train our model to learn scale-invariant features.
- This means that our model should learn the desired features or identify an object from a given image irrespective of the size, orientation, position and scale of the feature/object of interest.

2. CNN

- This is the main model (brain) that we have to train our “and” images on.
- A series of convolutional layers and pooling layers are used to construct this model.
- We intend to use a model with roughly the following structure (subject to change as per experimental results):

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000
lambda_2 (Lambda)	(None, 1000)	0

Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0

3. **Feature Maps**
    - The CNN generates a latent feature map of the input image.
    - These features are extracted to represent the original input image by using lesser, but significant, number of features.
    - In our case, the dimensions of the latent representation are: 512x14x14.
4. **Fully Connected Layer**
    - This layer can be used for the following applications:
        1. Classification Task
        2. Visual Explanation Task
        3. Textual Explanation Task
    - In our case, we have used this layer to perform the classification task such that the number of
5. **Individual Class Activations**
    - We get the probability distribution over all the classes and choose the one with the highest probability.
6. **Global Average Pooling**
    - We then take the score of the predicted class and weight each of the scores with the weights of the feature map layer.
    - This quantifies how much importance needs to be given to each of the classes (with a higher weight for most probable class).
7. **ReLU**
    - Only the class associated with the highest weighted probability score has a positive activation, rest all have negative ones.
    - Therefore, we need a ReLU activation function to filter out the positive activation and get rid of the negative ones.
8. **Grad-CAM Heatmap**
    - The selected activation helps us identify which was the predicted class according to which the heat map for that object (belonging to the predicted class) is generated.
    - The pixels that contributed to making the decision of predicting a certain class is included in the heat map representation.
9. **Output Image**
    - This heatmap is then overlapped with the original image to highlight the visualized part of the image that led to the prediction of the given class.

❖ EXPERIMENTAL SETUP

Following is the link for the colab notebook where we have setup the code: https://colab.research.google.com/drive/1zWM6dqLYmp1996J1iKY6iGm0uAbxTQI8

Functions Used:

1. **Pre-Processing Input Images** - ***preprocess_input()***
    - We have used the in-built Keras function called “preprocess_input()” that basically is meant to adequate the input image to the desired format that the model requires.
    - This is because some models use images with values ranging from 0 to 1, others from -1 to +1, others use the "caffe" style, etc and we have to modify our input images in accordance with the model requirements.
2. **Normalizing the image** - ***normalize()***
    - Normalization is one of the image pre-processing steps and is required to get a common ground (criteria) for all the pixel values for better representation of the images.
3. **Loading the input images** - ***load_image()***
    - This basically gets the input images that would be used for training and testing and performs some pre-processing on them.
    - This involves resizing and invoking the in-built keras function *preprocess_input()*
4. **Normalizing the image** - ***deprocess_image()***
    - The de-process image method normalizes the image and converts it to the grayscale version.
5. **Computing Gradients** - ***compute_gradients()***
    - The weights from the last layer are zipped together in this layer.
    - This function is called in grad_cam function while calculating the gradients.
6. **Grad-CAM** - ***grad_cam()***

- This is the main method where all the steps involved in Grad-CAM are executed.
- In this process, the output predicted class is decoded and the gradients are calculated with respect to the output layer of the model.
- By performing the global average pooling technique [i.e; taking the mean of grad values] we can calculate the new important weights of part of an image which was predicted by our model.
- We make the mask which is generally a blur effect on the image.
- This mask is superimposed on the image which was given to model to predict, to highlight only the predicted part and remaining part of the image is blurred.

❖ EVALUATION

- What makes a good visual explanation? We need a two-fold evaluation process:
    1. Model Evaluation
    2. Human Expert Evaluation

MODEL EVALUATION:

- Consider image classification – a ‘good’ visual explanation from the model for justifying any target category should be
  (a) class discriminative (i.e. localize the category in the image) and
  (b) high-resolution (i.e. capture fine-grained detail)
- As a result, important regions of the image which correspond to any decision of interest are visualized in high-resolution detail even if the image contains evidence for multiple possible concepts
- Localization approaches like CAM or our proposed method to use Gradient-weighted Class Activation Mapping (Grad-CAM), are highly class-discriminative.

HUMAN EXPERT EVALUATION:

- We design and conduct human studies to measure if Grad-CAM explanations help users establish appropriate trust in predictions from deep networks and show that Grad-CAM helps untrained users successfully discern a ‘stronger’ deep network from a ‘weaker’ one even when both make identical predictions

❖ EXPERIMENTS AND RESULTS

- Initially we trained our model with learning rate = 0.1 and optimizer as Adadelta where we are getting a validation accuracy of 77%.Our model was not able to predict the individual class.
- So we tweaked our hyperparameters by varying learning rate = 0.01 and optimizer as Adadelta, where we are getting a validation accuracy of 89% and the model was able to predict the individual classes of the given image to calculate the heatmap.
- So we further varied our hyperparameters with learning rate = 0.001 and varying the optimizer as Adadelta and SGD and also by varying the decay rate from 10e-5 to 10e-6 where we achieved a validation accuracy of 88% and 89.63%

❖ CONCLUSION

➢ We were able to visualize the localized heatmap for the image fragment with the use of weights generated the gradients generated by Grad-CAM.

➢ This heatmap can then be superimposed on the image fragment to better visualize the results which can be used to evaluate authentic or forged handwritten samples.

➢ Tuning the model with different settings of parameters would be the next phase of this project.

➢ We have summarized this content in the following blog: https://machinelearningfromub.home.blog/

❖ FUTURE WORK

➢ We plan to provide a User Interface for explanation tool where users will be able to get the textual as well as visual explanation for the task of handwriting verification.

➢ We can extend the image fragment analysis to a document verification one.

❖ REFERENCES

1. Grad-CAM ICCV paper: https://arxiv.org/pdf/1710.11063.pdf

2. Code reference: (https://github.com/PowerOfCreation/keras-grad-cam/blob/master/grad-cam.py)
3. Keras Representation of the code: https://github.com/jacobgil/keras-grad-cam/blob/master/grad-cam.py
4. CNN Visualization: https://mxnet.incubator.apache.org/versions/master/tutorials/vision/cnn_visualization.html
5. Grascam library functions: https://raw.githubusercontent.com/apache/incubator-mxnet/master/docs/tutorial_utils/vision/cnn_visualization/gradcam.py?raw=true https://towardsdatascience.com/understanding-your-convolution-network-with-visualizations-a4883441533b