Privacy Preserving Facial Recognition

Teresita Esver, Computer Science
Mentor: Dr. Tingting Chen
Kellogg Honors College Capstone Project

Motivation

- Facial recognition systems use Artificial Intelligence, raising data privacy concerns and potential human rights violations.
- The use of facial recognition by law enforcement is especially controversial.
- Major tech companies, including Microsoft and IBM, have stopped selling facial recognition technology to law enforcement agencies until federal laws regulate its usage.
- One approach to preserving personal privacy in facial recognition systems is to use Fully Homomorphic Encryption, or FHE[1].

Project Objective

- The objective of this project is to build and train an artificial neural network model to perform facial recognition, while preserving individual privacy.
- Using Microsoft CryptoNets,[2] the trained model will be converted into a privacy preserving model that uses FHE. FHE allows for mathematical operations to be done on encrypted data. The identity of the individual is not revealed during the facial recognition process.
- There are two main modules of this project. Module 1 is building the model, and Module 2 is condensing the trained model & applying FHE with Microsoft CryptoNets.
- This discussion is focused on Module 1, tackling the challenge of maintaining both accuracy and simplicity of the model in preparation for FHE application.

Methodology

- The first step was to build a neural network model with Python's neural network library, Keras.
- We trained the model with the VGGFace2[3] image dataset. To optimize the model's performance during feature extraction, we employed Multi-Cascaded Convolutional Neural Networks[4] (MTCNN) to detect the face from each training image, as shown in Fig 1.

Fig 1. MTCNN stages.
Credit: K. Zhang et al.

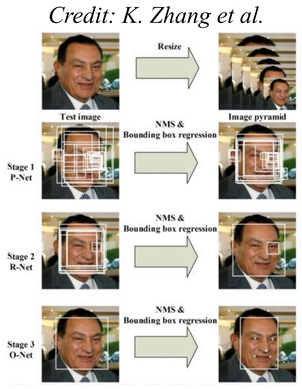
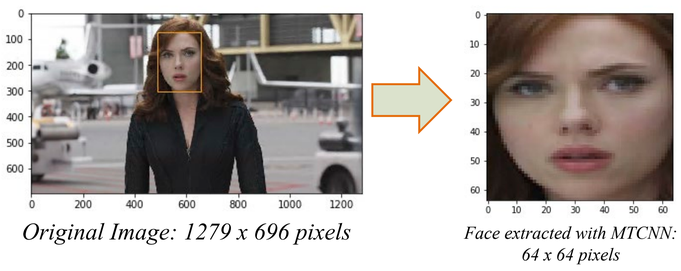


Fig 2. MTCNN extracts face and resizes original image.

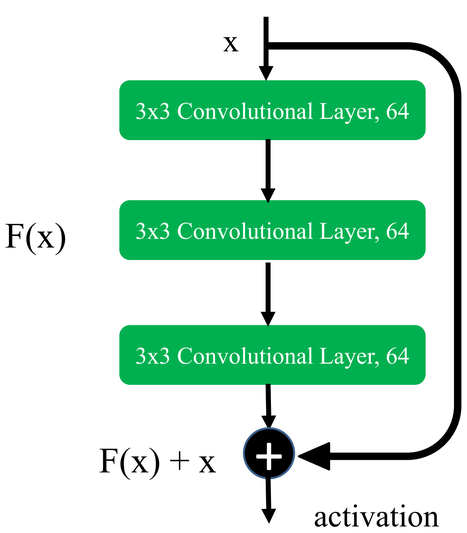


Original Image: 1279 x 696 pixels

Face extracted with MTCNN: 64 x 64 pixels

- Shown in Figure 2, MTCNN detects the face in the image based on the detected features, building a bounding box around the predicted location of the face.
- MTCNN also resizes the detected face to dimensions suitable for our model, which takes in image input of size 64 x 64 pixels.
- Facial Recognition is a complex task, requiring a deep neural network. We hypothesized that a Residual Neural Network[5] (ResNet) architecture would yield the best results.
- ResNet combats the vanishing gradient problem with deep neural networks by using shortcuts, called skip connections.
- Skip connections allow the network to continue updating the weights and thus continue learning.

Fig 3. A residual block in a ResNet.



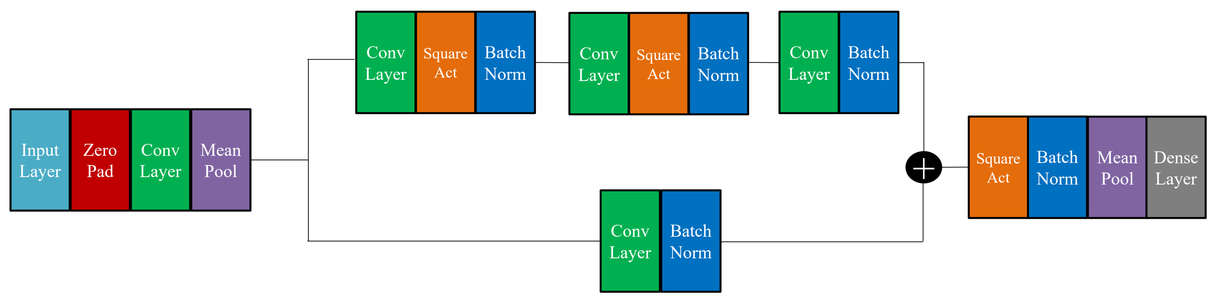
- Input x passes through a residual block, consisting of 3 Convolutional Layers, each with 64 filters of size 3x3.
- At the end of the residual block, input x from the beginning of the block is added to the mappings.
- The input passes through an activation function. The model learns from a residual mapping.

Experiments

We ran several experiments that led to the architecture portrayed in Figure 3 below, which has a test accuracy above 80%, while making a trade-off between accuracy and simplicity.

- We trained the model with 10 identities from the VGGFace2 image dataset. 90% of the data was used for training, and 10% of the data was used for testing, utilizing an 80/20 validation split. A total of 2110 samples were used for training and 528 samples for validation. The loss function is Categorical Crossentropy.
- ResNet utilizes the same types of layers as in a Convolutional Neural Network, featuring Convolutional Layers. We also used Batch Normalization layers.
  - Convolutional Layer: uses filters (of size n by n pixels) to extract features from an input image, outputting feature maps.
  - Batch Normalization Layer: standardizes and normalizes input.
- We replace reLU activation with square activation, x^2, in order to maintain simplicity for future FHE application.

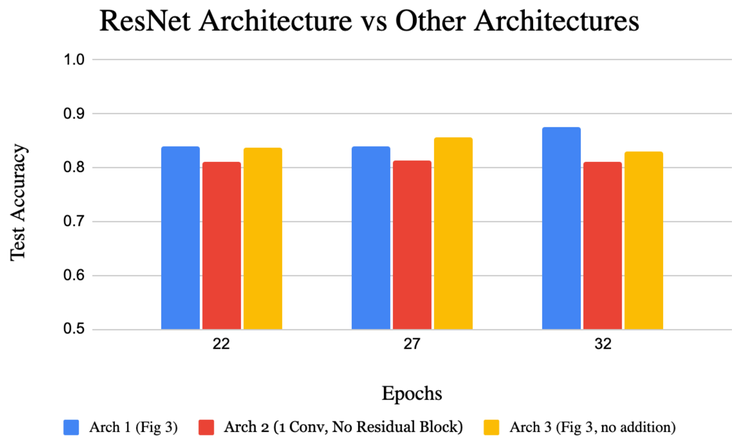
Fig 3. ResNet Architecture that yielded the best results on VGGFace2 with square activation.



- The Input Layer first takes in image input, followed by zero padding, a Convolutional Layer that has 64 filters of size 7x7 and a stride of 2. Average Pooling takes place, after which the image input diverges as it enters the residual block.
- The topmost blocks indicate the main path that the image takes, where there are a total of 3 Convolutional Layers. The first Convolutional Layer in this path has a stride of 2 and 64 filters of size 3x3. We then apply Square Activation and Batch Normalization. The input enters another Convolutional Layer with a stride of 1 and 64 filters of size 3x3, and same padding. Last in the main path is another Convolutional Layer with a stride of 1, 64 filters of size 1x1, and valid padding.
- The other path the input takes is reflected on the bottommost path, where there is one Convolutional Layer followed by a Batch Normalization Layer. The Convolutional Layer here has a stride of 2, 64 filters of size 3x3 and valid padding.

- We conducted multiple experiments (see Fig. 4) to solidify our choosing of the above ResNet Architecture (Arch 1). The test accuracies shown are the averages of 5 runs for each architecture.
- Architecture 2 (in red) serves as a base model. It has the Input Layer, Zero Padding, Convolutional Layer and Average Pooling as in Arch 1, but does not contain the residual block and final activation. After Average Pooling, it goes straight to the last Batch Normalization Layer and onward.
- Architecture 3 has the same layers as Architecture 1, but there is no addition and thus no skip connection. The input enters the main path, goes through the bottom path, into the final Square Activation & onward.
- Based on these results, Arch 1 is a viable option for later conversion to CryptoNets.

Fig 4. Comparing ResNet Architecture from above with others.



Based on these experiments, we found that the simple architecture from Fig. 3 best fits our goal to use Cryptonets and apply FHE.

Current & Future Work

At this point, we can proceed to Module 2 of the project, which is to the convert the model to Microsoft Cryptonets. The CryptoNets version of the model will only be tested, rather than trained & tested as in Keras. The goal is to achieve the same testing accuracy for the CryptoNets version of the model as the ResNet trained in Keras.

References

[1] Gentry, Craig. Fully homomorphic encryption using ideal lattices. In STOC, volume 9, pp. 169–178, 2009.

[2] Dowlin, Nathan and Gilad-Bachrach, Ran and Laine, Kim and Lauter, Kristin and Naehrig, Michael and Wernsing, John. "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy." https://www.microsoft.com/en-us/research/publication/cryptonets-applying-neural-networks-to-encrypted-data-with-high-throughput-and-accuracy/.

[3] Q. Cao, L. Shen, W. Xie, O. M. Parkhi and A. Zisserman, "VGGFace2: A Dataset for Recognising Faces across Pose and Age," *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, Xi'an, China, 2018, pp. 67-74, doi: 10.1109/FG.2018.00020.

[4] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," in IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.

[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.