REVISTA ELECTRO, Vol. 44 pp. 200-205, Oct 2022, Chihuahua, Chih. México
http:// electro.itchihuahua.edu.mx/revista
ISSN 1405-2172

EMBEDDED VISION SYSTEM WITH MULTIPLE CAMERAS FOR FACE RECOGNITION

Andrés Enrique Loya Domínguez, Isidro Robledo Vega*, Pedro Rafael Márquez Gutiérrez, Carmen Leticia García Mata, Alberto Pacheco González
Tecnológico Nacional de México / Instituto Tecnológico de Chihuahua
División de Estudios de Posgrado e Investigación
Tecnológico Ave. #2909, 10 de mayo, Chihuahua, México
Tel. +52(614)201-2014
andyeloyd@gmail.com, [isidro.rv, pedro.mg, carmen.gm, alberto.pg]@chihuahua.tecnm.mx
* Corresponding Author

ABSTRACT

This article presents the development of an embedded vision system with multiple cameras for monitoring access areas to a building and to carry on personnel identification via facial recognition. A Deep Neural Network (DNN) was trained on the VGGFace2 dataset using two different architectures, DFN-L and ResNet-50, as well as two different classification layers, fully connected and ArcFace. The behavior during training and discriminative performance of each combination of architecture and classification layer is compared to determine which is more appropriate for a continuously re-trainable network, observing that a ResNet-50 with regular fully connected classification layer fits the criteria better. This network was retrained on a dataset to identify a people which data was acquired locally, reaching 98.83% verification accuracy. This DNN was deployed on an Nvidia Jetson TX2 embedded system connected to a surveillance camera system to identify persons in the database.

Keywords: Face Recognition, Face Detection, Neural Networks, Biometric Systems, Deep Learning.

1. INTRODUCTION

In our current age, biometric systems, which use biological measurements to identify people based on their physical or behavioral characteristics have turned into the most reliable alternative to classical authentication methods, like the possession of an object such as a key or identification card, or knowledge of a password. A major advantage is that unlike traditional methods, biometric information cannot be lost [1]. Among biometric recognition techniques, facial recognition is highlighted for the ease of facial data acquisition, requiring low or even null effort from the subjects, and sufficing with the use of a common use camera, whereas the acquisition of other biometric information relies on specialized sensors [2].

Although progress has been made in facial recognition for several decades, the implementation of Deep Learning techniques, which are a branch of Machine Learning, has brought unprecedented success upon the field. These techniques involve the design of a deep neural network which is requires enormous amounts of training data, from which the network learns patterns to detect. As a result, it is possible to obtain systems capable of rivaling and even surpassing human capacities. Given enough computational resources, facial recognition systems can be implemented on plenty of computational systems. However, it can be preferred to run them on an embedded system. An embedded system contains one or several processors, and have specialized architecture fit to a specific task. Due to the custom fit nature of an embedded system, it generally performs faster and more efficiently than a common use computational system [3].

Highly developed as it may be, facial recognition is often marred by low quality images, such as those obtained with low resolution sensors or in badly illuminated scenes, as well as non-frontal images. Indeed, deviating from the narrow set of ideal quality images greatly degrades the performance of facial recognition systems, to the point where they can perform badly on images obtained "in the wild". Similarly, identifying faces with occlusions is a challenging problem.

In this article we present the development of an embedded vision system capable of performing face detection on image sequences obtained from a surveillance camera system as well as performing recognition of the detected faces, identifying the people registered on the training dataset. This system is capable of keeping a registry detailing the time and date of arrival of detected people in the monitored area. Figure 1 shows the process followed by the system.

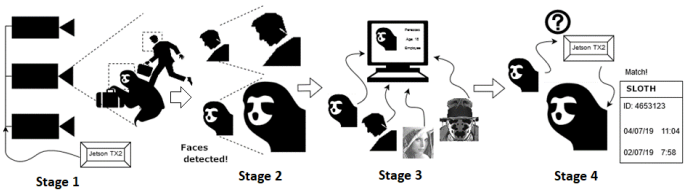

Figure 1. Overall embedded vision system process.

In the first stage the embedded vision system monitors a given area while connected to a surveillance camera system from which it acquires image sequences. In the second stage face detection is performed over these image sequences. During the

third stage recognition is carried out over detected faces to determine their corresponding identities based on a given dataset. Finally, during the last stage the found identities are registered.

2. IMAGE ACQUISITION

The embedded computing board is connected to a surveillance camera system. The cameras are Hikvision HiLook IPC-B620-Z, which feature high quality lenses with a wide field of view. Cameras are connected to a Local Area Network (LAN) via Ethernet cable, which also powers them via PoE (Power over Ethernet). The parameters of the cameras are show in Table 1.

Table 1. Hikvision HiLook IPC-B620-Z camera parameters.

Feature	Value
Interface	Ethernet
Power	PoE
Max. Resolution	1920x1080
Max. FPS	30
Scan type	1/2.8” progressive
Night Vision Range	30m
f-number	F1.4

To acquire image sequences from the cameras, a Real Time Streaming Protocol algorithm was implemented, making use of various functions from the OpenCV library. To access the cameras video stream, it is only necessary to specify their IP address in the algorithm running in the embedded computing board, where the face detection and recognition in performed.

3. FACE DETECTION

In computer vision, face detection can be accomplished through different means, but nonetheless, deep neural network (DNN) models often perform the best. In the developed embedded vision system face detection is performed in such a way, using a pre-trained DNN rather than training one from scratch.

When a pre-trained DNN is used in inference mode, that is, for doing the task it was trained to, it has to be fed with certain input data, which is processed through the network layers until it produces an output. In the context of face detection, the use of a DNN for such a purpose roughly follows the following steps:

1. The face detection DNN is fed an input image.
2. Different regions of the image are analyzed, where the DNN decides whether a face exists or not.
3. As a result, the DNN returns bounding boxes enclosing the detected faces, if any were found.

An important parameter is the scale of images, since a face of a very small size can get distorted and therefore be missed by the detector. This commonly happens when the faces in a scene are very far from the camera, appearing only as a very small set of pixels in the captured image.

In the realm of image recognition and detection, a particular type of neural network, the Convolutional Neural Network (CNN) reigns supreme, as the most widely used, and in most cases, also best performing of them. A CNN employs filters in its convolutional layers to generate activation maps, which similarly to the human eye react to visual stimulus. In a CNN the convolutional layers sample around local receptive fields to generate such a stimulus.

The DNN pre-trained model selected to perform face detection is the Multitask Cascaded Convolutional Neural Network (MTCNN) [4], which works by joining three different networks in sequence to perform the following tasks:

1. Obtaining candidate windows through the analysis of a simple CNN [5].
2. Refining of candidate windows, by eliminating face-free windows with a more complex CNN.
3. Further refining of the windows by eliminating more face-free windows with an even more complex CNN.

MTCNN reaches real time performance when used in 640x480 images, and features several parameters that can be tuned to filter out faces of small resolutions and to define confidence thresholds, among other things. It is capable of detecting faces as small as 20x20 pixels.

4. FACE RECOGNITION

Similarly, to face detection, face recognition can be performed through many different techniques, but most of them pale to in comparison to DNNs. Differently from face detection, face recognition is performed in this embedded vision system not by using a pre-trained DNN, but by one trained from zero.

4.1. Network Architecture and Loss Functions

Although many of the state of the art DNNs have focused on architectures with a large amount of parameters, sometimes reaching the billions, the designed embedded vision system is oriented towards real time performance, or at least a performance close to it. Since a big architecture implies a higher amount of calculations for the network to produce an output, the largest of DNNs are of little interest when implemented on embedded devices of limited resources. In general, the leaner the architecture the faster it will compute. With such a premise in mind, particular attention was given to lightweight DNN architectures found in literature.

Another specific concern was the performance in non-frontal face positions, since the further a face deviates from frontal view, the worse the networks predicts its identity. Given that in the developed system the image sequences are acquired through a surveillance camera system in a mostly uncontrolled scenario, the angle of the faces with respect with the camera is expected

to vary wildly. Therefore, proposals for tackling positional variance were thoroughly studied.
An interesting network architecture is the Deformable Face Network (DFN), which has the peculiarity of featuring a "deformable module" consisting of a deformable convolutional layer [6]. In contrast to regular convolutional layers, a deformable convolutional layer is capable of adapting its receptive field during training to better model certain geometric transformations. The rationale behind this design choice is that faces follow semi-rigid deformations, such as the change of position. A regular convolutional layer would model transformations only following a rigid square pattern.

DFN demands to be trained by inputting data in pairs of images of the same class. It also makes use of three novel loss functions: displacement consistency loss (DCL), identity consistency loss (ICL) and pose triplet loss (PTL), which are used as regularization terms to the Softmax loss, which computes the output of the network, and is a common feature in many different DNNs. DCL attempts to learn the displacement fields, to produce a consistent output, while ICL constraints the geometric distance between the output features produced by each pair of input images to ensure similar identities produce similar outputs. PTL was not considered for further examination.

Although the Softmax loss is popular, further advances have been made through margin loss. Margin loss rely on the geometric representation of each training class on the network. As they are dispersed across the representational hyperspace, a margin is imposed between each pair of classes, squishing the classes into narrower regions and further separating them from each other. This serves to define a clearer decision boundary, which separates each class. This is more clearly seen in the Additive Margin Softmax loss[7] as shown in Fig. 2.

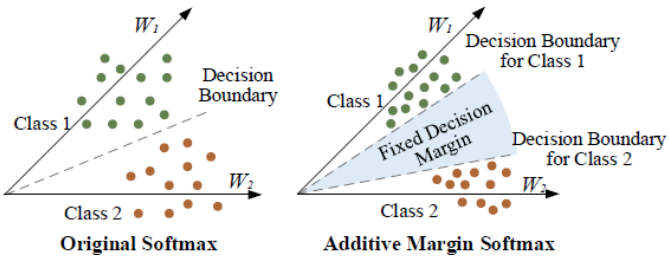

Figure 2. Softmax loss and Additive Margin Softmax loss.

Particular interest was given to the ArcFace loss, a margin loss which makes use of two parameters to scale the representational hyper-space and to impose a margin in the angle measured between each class center in the hyper-space [8]. These parameters, known as s and m, scale the total radius of the hypersphere enclosing the representations and the angular margin between each class center, respectively. DNN training is highly influenced by these values. Unfortunately, there is not much of a theoretical background for the determination of appropriate parameter values, and some of the better performing values have been found in a purely empirical way. Generally, they are set to m=0.5 and s=64. An alternative to calculate s give in [9] is based on the number of classes to distinguish from:

$$s = \sqrt{2} * \ln(C - 1) \quad (1)$$

4.2. DNN Training

For comparing the effectiveness of the architectures and loss functions of interest, some DNNs were trained, featuring a few differences between each other. DFN was implemented on its lighter variant DFN-L, and compared against a ResNet-50 [10]. In the designed embedded vision system, face recognition relies on the identities the DNN has been trained on, and it will rely on re-training the DNN to add a new identity. Therefore, particular concern was given to the training time of the network.

All of the DNNs were trained on the VGGFace2 dataset, which was created with the explicit purpose of collecting high quality labeled images of faces in a wide variety of poses and of featuring a diverse spread of ages and ethnical identities [11]. VGGFace2 is composed of 3.31 million images belonging to 9131 identities, with an average of 362.6 images per identity. Alignment was performed on each image of VGGFace2, and each of the images was cropped to a resolution of 122x122.

Since deep learning framework of choice is TensorFlow with Keras used in tandem, the TensorFlow exclusive format TFRecord was used for speeding up the training time by applying it to the training datasets. A TFRecord essentially converts each image in a dataset to a binary format which can be mapped back to image format through a previously defined parsing function. Each of these binary sets of data is gathered into a single file, which can then be split to convenience. The advantage of using TFRecords is that if we use the images separately, each of them would need to be opened separately, but since a TFRecord shard contains information of many images, we can greatly reduce the performance impact of opening them [12].

To monitor how much the capabilities of a network improve while being trained, it is common to evaluate their performance each time an epoch passes. An epoch is completed each time the networks has run through the entirety of the training dataset one time. The evaluation is usually performed by using a validation dataset, which is a small split from the original training dataset, and is kept for this purpose only. In other words, the DNN is not trained on the validation dataset and is used only for performance evaluation.

For the DFN-L training, the weights given to the ICL and DCL regularization losses were 0.001 and 0.01, respectively. The momentum and learning rate were continuously modified through a One-Cycle Policy (OCP) training criteria of 20 epochs following the criteria used in [13]. As shown in Figure 3, DCL behaved in a marked parabolic way, increasing during the first half of the training, until finally starting to get low later on. After these 20 epochs, the DFN-L trained on VGGFace2 reaches a validation accuracy of around 82% as show in Figure 4. Then, we performed training for 4 more epochs with a regular training regime using a Nesterov moment of 0.9 and a reduced learning rate of 0.001, the model finally converges with a validation accuracy of 87%, higher than those achieved with other unconstrained data datasets in previous research.

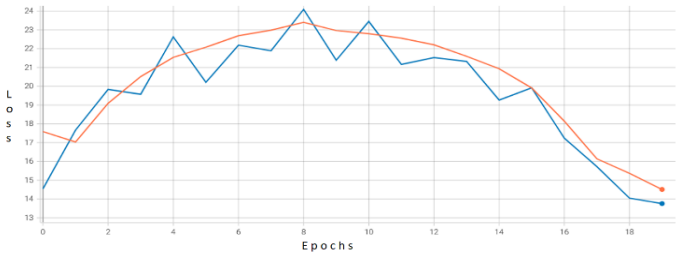

Figure 3. DCL loss during DFN-L training (blue) and validation (orange).

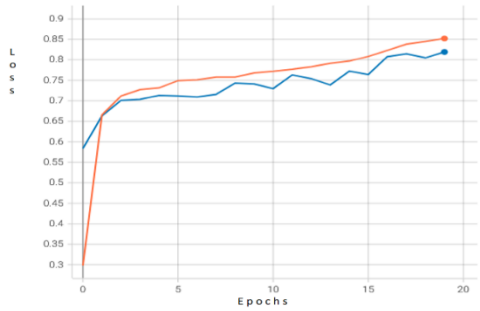

Figure 4. Training (blue) and validation (orange) accuracy during 20 epochs of DFN-L training.

It became evident at this point that although the achieved accuracy on DFN-L was unprecedented, it still ended up short for a biometric system. Although it showed promise, in its current form it's too dependent on being trained with highly constrained data, which is out of scope for the developed embedded vision system. Focus was then shifted towards the ResNet-50 architecture, and the impact of using an ArcFace loss function. ArcFace constraints the representational hyperspace through its parameters s and m, and is implemented as a classification layer used as an alternative to fully connected layers. Since it only affects the last layers of the architecture, experiments were made by performing transfer learning of a ResNet-50 previously trained on VGGFace2. This process involves stripping the last network layer, the fully connected classification layer, and replacing it with a new classification layer. Afterwards, the rest of the model is frozen so that the weights of layers other than the classification layer are not further modified. Training is then resumed.

Since the formula in Eq. 1, for setting the Arcface hyperparameter s, depends on the number of classes, experiments were conducted to observe how much the networks training is affected varying the number of classes. They consisted in training the networks using two different subsets of VGGFace2: one with 500 identities, and other with 20 identities. In regards to the second one, 4 more identities were added, corresponding to subjects whose data was collected locally following an uncontrolled methodology to better approach the distribution within VGGFace2. A data augmentation process was applied to the images to obtain a total number of images for each identity similar to the average of 362.6 per identity on VGGFace2. The different networks trained using the ResNet-50 transfer learning approach and their obtained results are displayed in Table 2. All of them followed a regular training regime, using Stochastic Gradient Descent with a learning rate of 0.1 and momentum of 0.9.

Table 2. ResNet-50 DNNs trained through transfer learning.

DNN	Classification Layer	Number of Classes	Epochs	Validation Acc. (%)
1	ArcFace	24	6	98+
2	Fully connected	24	25	98+
3*	ArcFace	500	35	55
4	Fully connected	500	9	99+

*DNN #3 was not trained up to convergence, since by the time it reached 35 epochs only obtained a validation accuracy of 55%. This is enough to show that for a high number of classes, ArcFace trains to slowly in comparison to traditional fully connected classification layers.

The slow training speed of ArcFace with a high number of classes is a significant handicap for a network that need to be constantly re-trained, as stipulated on the designed embedded vision system. In addition, no direct benefits over a regular fully connected classification layer were observed. Due to these reasons, ArcFace was considered a downgrade to a fully connected layer, and will not be used.

5. PROGRAM PIPELINE

The developed program consists of a) acquire images from surveillance cameras, b) detecting faces on them, c) recognizing the detected faces, and e) registering date and time of first detection. More specifically, after detecting one or more faces through MTCNN, we obtain a region of interest for each of them in the form of bounding box coordinates. Using those coordinates, the program crops that specific region from the input image and passes it to the ResNet-50 trained for face

recognition. This network will then assign the face to a class, which correspond to an identity, by outputting a probability of it belonging to each class. The sum of these probabilities is always 100%. If the probability for one class is above a preset confidence threshold, the recognition is taken as valid, representing a potential sighting. Next, a routine similar to the debouncing of a button is performed, which requires that we reach a similar result in the pipeline recognizing the same face for a given number of frames, which were set to a default of 8. During this trial period, an adjustable number of failed recognitions is allowed, set to a default of 3. A detailed flow diagram of the system is shown in Figure 5. Since most DNNs don't have a perfect 100% accuracy, just like the ones used, performing this filtering allows the avoidance of the small incidence statistical inaccuracy.

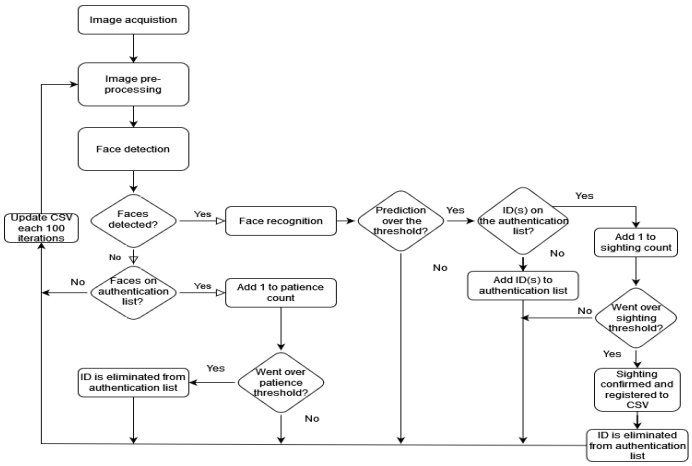

Figure 5. Full program flowchart.

6. DEPLOYMENT TO EMBEDDED SYSTEM

The face recognition system was deployed to an embedded system. Among the many embedded systems available on the market, just a few of them are oriented towards the deployment of software for computation intensive tasks such as deep neural networks for computer vision applications. The Nvidia Jetson embedded computing board series, which integrates a CPU, GPU and other circuits in a single system, was created by Nvidia for this explicit purpose. The developed program was deployed to an Nvidia Jetson TX2 embedded device [14], whose specs are displayed in Table 3.

Table 3. Nvidia Jetson TX2 specifications [14].

Feature	Jetson TX2
CPU	4-core ARM Cortex A57 @ 2 GHz, 2-core Denver2 @ 2 GHz
GPU	256-core Maxwell @ 1.3 GHz
Memory	8 GB 128-bit LPDDR4, 58.3 GB/s
Disk Storage	32 GB eMMC 5.1
Tensor Number	--
Video Codification	(1x) 4Kp60, (3x) 4K360, (4x) 1080p30, (8x) 1080p30
Video Decodification	(2x) 4Kp60, (4x) 4Kp30, (7x) 1080p60
USB Ports	(1x) USB 3.0 + (1x) USB 2.0

For further streamlining, the DNN used for face recognition was converted from a TensorFlow model to a TensorRT model. The latter is an SDK oriented to DNNs used in inference mode, making extensive use of the CUDA library to create high performance GPU-accelerated applications. Also, in comparison to TensorFlow, TensorRT has a smaller memory consumption, which aids in maintaining a lean resource footprint.

With the face recognition software deployed in its entirety to the embedded device, it was connected to the surveillance camera system described in section 2 through Ethernet connection. Although, to register sighted people, no peripherals aside from the cameras are needed, it is necessary to interface with the embedded system in order to run the program. This is achieved via a monitor, keyboard, and mouse connected through physical ports in the Nvidia Jetson TX2 embedded board.

7. EXPERIMENTS AND RESULTS

Two programs were written: one which performs the passive registering of the recognized identities in the image sequences, and one intended for demonstrating the internal process of the face recognition software. In the later, instead keeping a registry, it displays on screen the detected faces and their corresponding predicted identity on top of the original input image. The two programs will henceforth be referred to as the registration program and the demonstration program, respectively.

The face recognition system based on ResNet50 architecture provides an accuracy of 98.83% on the validation data set, according to the experiments shown in Table 2. When testing with data acquired in real time, the system correctly recognized the identities of the persons in the database, having only one false positive when the face detected was in an almost lateral view. Figure 6 shows a person recognized from the two camera images at different times and Figure 7 show the recognition of two persons from two camera images, both using the demonstration program. Figure 8 shows some records in a CSV file made by the registration program.

Figure 6. Face recognition from two cameras.

Figure 7. Recognition of two subjects from two cameras.

	Standard	Standard	Standard
1	Name	Last Date	Last Time
2	"Andres"	2022-01-03	12:42:34.854808
3	"Laura"		
4	"Abue"		
5	"Gera"	2022-01-03	12:37:38.539832
6	"14th_Dalai_Lama"		
7	"AYAMI"		
8	"Abdullah_II_of_Jordan"		

Figure 8. CSV file with sighting records.

While using just one camera, running either the demonstration program or the registration program, the system achieves a speed performance of around 5 FPS while using up 2 GB of memory in addition to the passive memory use of the embedded board, out of the 8 GB available. This accounts for around 60% of the total memory. Using two cameras drops this speed performance to around 4 FPS, with an only marginally higher memory consumption.

8. CONCLUSIONS

A prototype of an embedded vision system was developed, capable of detecting faces in image sequences obtained from surveillance cameras, and of recognizing the identity of people registered in a dataset. The system keeps a registry of the people accessing a monitored area. The software developed consisted of a face detection module based on MTCNN and a face recognition module implementing a DNN, different architectures and loss functions such as the DFN-L network and ArcFace loss functions were compared through training experiments showing long training periods. They were discarded in favor of the ResNet-50 architecture, which proved significantly faster to re-train. ResNet-50 was pre-trained on the VGGFace2 dataset and later on trained through transfer learning on a dataset created with some identities from VGGFace2 and added new identities collecting data locally. According to the experiments carried out, the face recognition system based on ResNet50 architecture provided an accuracy of 98.83% on the validation data set.

A filtering algorithm was written to discard false positives obtained through the face recognition software across multiple frames, ensuring a proper registering of the identities recognized. While deployed on an Nvidia Jetson TX2 embedded board, the system used up only around 60% of its resources and kept a near real-time performance while using two cameras, allowing more cameras to be connected.

9. REFERENCES

[1] A. Jain, R. Bolle, S. Pankati, *Biometrics: Personal Identification in Networked Society*. Norwell, MA: Kluwer Academic Publishers, 1999.

[2] L. Masupha, T. Zuva, S. Ngwira, O. Esan. “Face recognition Techniques, their Advantages, Disadvantages and Performance Evaluation”, 2015 International Conference on Computing, Communication and Security (ICCCS)

[3] P. Viola, M. Jones, “Rapid object detection using a boosted cascade of simple features”, Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, HI, USA, 8–14 December 2001

[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] Y. LeCun and Y. Bengio, ‘‘Convolutional networks for images, speech, and time series,’’ in The Handbook of Brain Theory and Neural Networks, A. A. Michael, Ed. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258.

[6] M.He, J. Zhang, S. Shan, M. Kan, X. Chen, “Deformable face net for pose invariant face recognition”, in *Pattern Recognition*, 107113. doi: 10.1016/j.patcog.2019.1071, 2019.

[7] F. Wang, J. Cheng, W. Liu and H. Liu, "Additive Margin Softmax for Face Verification," in *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926-930, July 2018, doi: 10.1109/LSP.2018.2822810.

[8] J. Deng, J. Guo, N. Xue, S. Zafeiriou, “ArcFace: Additive Angular Margin Loss for Deep Face Recognition”, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4690-4699

[9] X. Zhang, R. Zhao, Y. Qiao, X. Wang, H. Li, “AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations”, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10823-10832

[10] K. He, X. Zhang, S. Ren, J. Sun. "Deep Residual Learning for Image Recognition". Proceedings of the IEEE Computer Vision and Pattern Recognition, 2016. p. 770-778.

[11] 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)*, 2018, pp. 67-74, doi: 10.1109/FG.2018.00020.

[12] “TFRecord and tf.train.Example”, TensorFlow Core, 2021. [Online] Available at: https://www.tensorflow.org/tutorials/load_data/tfrecord [Last Accessed: 12/22/2021]

[13] L. N. Smith, " A disciplined approach to neural network hyper-parameters: Part 1 --learning rate, batch size, momentum, and weight decay”. arXiv preprint arXiv:1803.09820, 2018.

[14] “Jetson TX2 Module”, NVIDIA Developer, 2018. [Online] Available at: https://developer.nvidia.com/embedded/jetson-tx2 [Last Accessed: 11/27/2019]