Integrating Location Information As Geohash Codes in Convolutional Neural Network-Based Satellite Image Classification

Mahara, Arpan; and Rishe, Naphtali

Abstract: ***In the past few years, there have been many research studies conducted in the field of Satellite Image Classification. The purposes of these studies included flood identification, forest fire monitoring, greenery land identification, and land-usage identification. In this field, finding suitable data is often considered problematic, and some research has also been done to identify and extract suitable datasets for classification. Although satellite data can be challenging to deal with, Convolutional Neural Networks (CNNs), which consist of multiple interconnected neurons, have shown promising results when applied to satellite imagery data. In the present work, first we have manually downloaded satellite images of four different classes in Florida locations using the TerraFly Mapping System, developed and managed by the High Performance Database Research Center at Florida International University. We then develop a CNN architecture suitable for extracting features and capable of multi-class classification in our dataset. We discuss the shortcomings in the classification due to the limited size of the dataset. To address this issue, we first employ data augmentation and then utilize transfer learning methodology for feature extraction with VGG16 and ResNet50 pretrained models. We use these features to classify satellite imagery of Florida. We analyze the misclassification in our model and, to address this issue, we introduce a location-based CNN model. We convert coordinates to geohash codes, use these codes as an additional feature vector and feed them into the CNN model. We believe that the new CNN model combined with geohash codes as location features provides a better accuracy for our dataset.***

Index Terms: ***CNN (Convolutional Neural Network), Data Augmentation, Geohash Code, Satellite Image, Transfer Learning***

1. INTRODUCTION

THE classification of remotely sensed data has numerous practical applications, including forest fire detection, landslide detection, and environmental monitoring. In recent years, several machine learning and deep learning algorithms, including but not limited to K-Nearest Neighbor (KNN), Random Forest (RF), Support Vector Machine (SVM), and Neural Networks (NNs), have been applied to the classification of remotely sensed data. In the Deep Learning field, CNNs have demonstrated the capability to learn complex models [1]. One of the key reasons for CNNs' success is their ability to extract features automatically, which greatly benefits researchers in achieving generalized and efficient classification. Comprehensive reviews of various models, architectures, and classifications related to CNNs can be found in references [1]–[3].

In general, image classification is performed based on pixel-wise feature extraction and assigning them to certain classes. Mnih proposed a CNN architecture for aerial image classification using a patch-based framework[4]. In that paper, the CNN network outputs a dense classification patch rather than a single categorical value. As a result, the patch-based CNN architecture increases the number of unproductive trainable parameters, potentially leading to inefficiencies in classification. To provide a solution to this issue, Maggiori *et al.* [5] proposed a fully convolutional architecture that only incorporates the convolution and deconvolution norms of CNN, producing classification maps that can be used for satellite image classification. In [5], the authors have created a more efficient CNN architecture, but their focus was on binary classification with only one class, i.e., buildings. The authors have not addressed the importance of using image location to enhance classification accuracy. The CNN architecture we use in this paper is based on the architectures described in [4] and [5], and we focus on multi-class image classification by integrating the location concept. In [6], coordinates were integrated into CNN to enhance remote sensing image classification. During the training phase, they directly fed spatial information, such as longitude and latitude, as an additional feature to the CNN for feature extraction. Similarly, Tang et al. [7] proposed a GPS encoding idea that incorporates location information into CNN for extracting features and improved image classification. They represented location as a

Manuscript received March 24, 2023.

Arpan Mahara is at the Knight Foundation School of Computing and Information Sciences, Florida International University, Miami, FL, USA (e-mail: amaha038@cs.fiu.edu)

Naphtali Rishe is at the Knight Foundation School of Computing and Information Sciences, Florida International University, Miami, FL, USA (e-mail: rishe@cs.fiu.edu)

Correspondence email is amaha038@cs.fiu.edu

binary code, with each bit corresponding to a specific geographic location. They devised a method for creating a set of grid cells covering the Earth's geographical area, primarily focusing on regions within the United States.

In the present work, we first downloaded satellite images of Florida using TerraFly Map's raster API, which incorporates a predefined tile system utilizing the Microsoft Bing projection. The images are all 256 * 256 pixels and have three color channels (Red, Green, and Blue). We have grouped the images into four different classes: Building, GreeneryLand, House, and WaterResource. Here, by a "house" we mean a structure of 1-2 stories, and by a "building" we mean a structure of 3 or more stories. We have developed our CNN architecture based on the idea mentioned in [4] and [5]. However, CNNs require large datasets to learn features and make efficient predictions, and our CNN may not be able to generalize efficient classification from manually-collected datasets due to the lack of a large number of images. To improve the efficiency of our classification, we first use data augmentation presented in [8]. Then, we adopt transfer learning strategies presented in [9] to extract features using pre-trained models, such as VGG16 and ResNet50. Finally, we enhance the CNN's feature set by converting each longitude and latitude to geohash codes and feeding them as extra features. Geohash is a process that converts coordinates into strings of data, which are easy to handle; more information on geohash codes can be found in [10]–[12]. We then evaluate the accuracy of our model with these additional features.

The paper is structured as follows. In Section 2, we describe the mechanism of CNN and how we have prepared the dataset. In Section 3, we propose a CNN architecture and analyze the shortcomings of the lack of a large dataset. In Subsection 3.1, we set up a transfer learning architecture to obtain an efficient classification model. In addition, we integrate coordinates as geohash codes into our model. Section 4 presents the computational results achieved from all the models, including the results obtained after the integration of location information. Finally, we summarize our findings and outline future research directions in Section 5.

2. CNN Introduction and Dataset

CNNs are a special type of neural networks that have been invented to mimic the mechanism of human brain for identifying or recognizing objects. They contain numerous interconnected neurons, each of which responds only to their own receptive field. The interesting part of the neurons in CNNs is that they possess the ability to automatically extract features from an image. In a CNN, each neuron undergoes input and output procedures to learn the pattern of the model. The common mathematical interpretation of the neural operation to obtain an output 'o' can be expressed as follows:

$$o = \sigma\left(\sum_{k=1}^{n} w_k \,.\, x_k + b\right) \quad (1)$$

where σ is an activation function that helps the CNN to learn an intricate pattern by encompassing non-linearity in the output. Similarly, x_k and w_k are k^{th} input and k^{th} weight, respectively, and b denotes a scalar parameter added to each output, which helps the CNN to extract complicated patterns from data. Biases should be carefully addressed; otherwise, they may lead to overfitting or underfitting in the model.

In general, the CNN architecture has three different layers: a convolutional layer, a pooling layer, and a connected layer. In the convolutional layer, the dot product between the kernel and the input image is calculated by sliding a filter over the image. This aids the architecture in extracting features from the images in the dataset. The sliding of the filter around the image can be controlled with a specific stride size. Let's say we have an image of dimension D*D with C channels. We define the size of the stride as S, the size of the kernel or filter as K, and X as the amount of padding to maintain the same size of images in both the input and output sectors. The output of the convolutional layer can be stated as follows:

$$C_{out} = \frac{D-K+2X}{S}+1 \quad (2)$$

Once the output is calculated, it is passed through an activation function. A pooling layer is applied in the CNN in order to deduct trainable parameters and balance the computation, which serves as an efficient feature extraction by reducing the size of the output map obtained from the convolutional layer. A fully connected layer simply flattens the output obtained from the previous layer, which helps to connect the obtained features to the labels in the given model.

As mentioned above, we used the TerraFly Map's raster API (which uses the Microsoft Bing projection) to download the images. We use the TerraFly Map to determine the XY tile coordinates for specific regions within Florida, keeping the zoom level constant at 19. After determining those coordinates, we pass the values to the Raster API's URL, and then we use web scraping to download the images. Since we aim to integrate the location feature into our CNN model, we need to prepare a dataset of satellite images that also have associated coordinates. To achieve this, we converted each XY tile coordinates obtained from the map to longitude and latitude by using the following procedure:

A = X tile, B = Y tile coordinates (i)

$$pixelA = A * 256 + 128 \quad \text{(ii)}$$

$$pixelB = B * 256 + 128 \quad \text{(iii)}$$

$$sizeofMap = 256 * 2^{zoom} \quad \text{(iv)}$$

$$normA = (pixelA / (sizeofMap)) - 0.5 \quad \text{(v)}$$

$$normB = 0.5 - (pixelB / (sizeofMap)) \quad \text{(vi)}$$

$$Latitude = 90 - \left(\frac{360}{\pi}\right) * \tan^{-1}(\exp(-2\pi * normB)) \quad \text{(vii)}$$

$$Longitude = 360 * normA \quad \text{(viii)}$$

In our model, we use a specific zoom value of 19. To calculate the geohash code, we use the values of the latitude and longitude obtained from equations (vii) and (viii), as described in [12]. We use the Python Geohash Library to convert the latitude and longitude to geohash codes. In our final step of dataset preparation, we map each geohash code to the right images by using a Python dictionary. The keys of the dictionary are the filenames, and the values are the corresponding geohash codes.

3. *The Proposed Architecture*

Our CNN architecture utilizes ideas from [4] and [5]. We apply convolutional layers that incorporate both convolutional and deconvolution operations, as described in [5]. We flatten the multi-dimensional tensor into a single-dimensional tensor output and apply the dense layer principle the output, as suggested by Mnih [4]. We feed the fully connected layer of the images into the CNN to extract the feature map, which is used for classifying the images according to their given labels. Our CNN architecture differs from the one presented in [6], as we focus on extraction that is capable of detecting features in the images, rather than extracting the spatial features of pixels in the images. As we can observe in Figure 1, the CNN architecture has three convolutional layers and three max pooling layers. In each max pooling layer, we downsample the dimension of each input map by a factor of 2, resulting in a feature map of size 32*32. Downsampling is a common approach in neural networks to reduce memory usage during computation and to enable high-level feature extraction [13]. We flatten the resulting feature map by applying a flatten layer, which transforms it into a one-dimensional array of size 65,536. We then apply two separate dense layers followed by a Softmax activation function. The final dense layer has 4 units, as our model has 4 classes of satellite images and the probability distribution is over those 4 classes.

In the first stage of our image classification procedure, we use a satellite image dataset that excludes geohash codes. We split the dataset into a training set, a testing set, and a validation set, with 80%, 10%, and 10% of the full dataset, respectively. We have experimented with our model using various numbers of epochs and batch sizes, and have determined that using 60 epochs with a batch size of 32 produces the best results. In general, researchers tend to choose an optimal number of epochs to achieve good accuracy in complex models and prevent the model from overfitting. A lower accuracy in the testing set indicates that the model is overfitting. One reason for this overfitting is the lack of a large amount of data in our model, as we only had 300 images in each class, with a total of 1200 images. CNNs require a large dataset to extract complex features and provide better accuracy in image classification [13].

To address this problem, we have used data augmentation strategies of deep learning, as presented in [8], [14], and [15]. In terms of images, data augmentation involves increasing the size of the dataset by applying variations, such as rotating images, changing the visual effects, etc., to the existing images [14]. To increase the size of the dataset, we have applied random horizontal flipping, random rotation with approximately 8.62 degrees, and random zooming of 20% scale. The data augmentation has helped to address the problem of overfitting, but we have concluded that we can further increase the overall accuracy of our dataset by training our model using a pretrained model, such as VGG16 and ResNet50, with the concept of transfer learning. In the following Subsection 3.1, we provide details on how we use transfer learning in our model to improve overall accuracy.

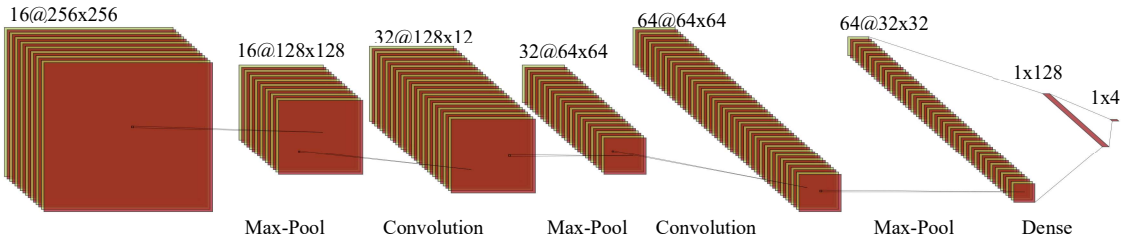

Figure 1: An illustration of the architecture of the CNN used. The template of the image has been obtained via https://alexlenail.me/NN-SVG/LeNet.html.

3.1 Applying Transfer Learning

Transfer learning is a way to use a pretrained model in a different but related model to solve the problem of the lack of abundant data to extract effective features and reduce the time required for training the dataset [16]. Our idea on integrating Transfer Learning is based on [9] and [17], and we have selected VGG16 and ResNet50 as the two pretrained models for our experiment. VGG16 is one of the most widely used deep neural network architectures; it consists of 13 convolutional layers and 3 dense layers, and the model has been trained on the ImageNet dataset [18]. Similarly, ResNet50 is another widely used deep neural network trained on the ImageNet dataset, consisting of 50 layers in total; it enables the network to assimilate residual functions rather than underlying mappings [19].

To set up the model using the transfer learning idea, we first remove the final connected layer of the VGG16 model. Then, we use the pretrained weights, and we set up the desired input shape to 256*256*3, the same shape that matches the input shape of the images in our original dataset. We freeze all the pre-trained layers and use only pre-trained weights to extract features, training the two new dense layers to predict new images in the dataset. The output of the flatten layer obtained from the pretrained model is passed to the first dense layer with 256 neurons, followed by the Relu activation function. In addition to this, we apply the Dropout function to the output from the dense layer to prevent overfitting in our model. Thereafter, we apply the final fully connected layer with 4 output nodes to obtain the probability distribution among 4 classes to predict the images followed by the Softmax activation function. We follow the same procedure when using the ResNet50 model. Once both models were ready, we experimented with them in our dataset. However, we have found misclassification in some of our data, which was further hindering the accuracy. To improve the accuracy, we integrate location coordinates in our image classification model in Subsection 3.2.

3.2 Integration of Location as Geohash Codes

Our goal is to increase the accuracy of satellite image classification in the downloaded dataset by integrating location information. We have decided to use geohash codes obtained from the conversion of latitude and longitude values. Geohash is a type of data structure used with spatial data that provide an encoding of latitude and longitude [20]. We are motivated to use geohash codes because locations with long common geohash prefixes are generally located nearby each other [20]. Our dataset contains satellite images downloaded within Florida, and there is a correlation between geographical location and image content. Images of houses and buildings are in two different classes, and some of these images might be misclassified if the model only considers visual characteristics because building images and house images captured from satellites have some visual similarity. In our dataset, two images of houses or buildings tend to be nearby each other as they have been downloaded by specifying the tiles coordinates. We believe that we can exploit this idea in our model by using geohash codes and prevent the misclassification of data.

We have experimented with the location concept by incorporating geohash codes into the VGG16 pretrained model. We have converted each geohash code into a floating-point value since neural networks typically deal with numerical values rather than strings. Next, we add a new input layer for the geohash code and concatenate the flattened layer containing the weight features of VGG16 with the geohash codes, as shown in Figure 2. We then apply the same dense layers noted in Subsection 3.1 to extract the features that assist in the prediction of new images. We follow the same procedure of concatenation geohash code with the output layer in ResNet50.

Finally, we integrate location information, i.e., geohash codes, into our CNN architecture. We concatenate the feature map induced by applying 3 convolutional and 3 pooling layers with geohash codes to obtain a combined feature vector. We then follow the same procedure as noted above and apply a flatten layer and a dense layer, respectively, to the combined feature vector. We have experimented with our models using 60 epochs and a batch size of 32 to obtain accuracy.

4. Computational Results

We have sequentially tested all the models, starting from the CNN architecture that only extracts features from the image without concatenating the geohash codes. Our intent is not just to check the accuracy in the dataset but also to analyze whether the model is overfitting by checking how well it performs on unseen image data. We use the Top-1 accuracy metric to check accuracies on all the models. Our CNN architecture yields an accuracy of 0.9244 on the training set but only 0.8842 on the testing set, indicating overfitting due to the limited size of the dataset. To address this issue, we apply data augmentation to the dataset, and our CNN architecture can generate approximately 0.9185 accuracy on the testing dataset and 0.9253 on the training set. Even though data augmentation helps to increase the dataset, it still lacks the power to generalize efficient feature extraction. So, we utilize transfer learning by using pretrained models, VGG16 and ResNet50, for efficient feature extraction that could be used to obtain better accuracy in our dataset. Having tested these models on all the datasets, we achieve an accuracy of approximately 0.9456 on the testing set and 0.9529 on the

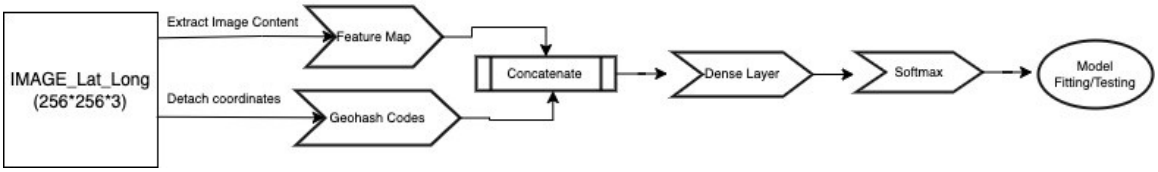

Figure 2: An illustration of the architecture obtained by integrating geohash codes into the CNN (including pretrained models) architecture.

training set for VGG16, as well as approximately 0.9516 on the testing set and 0.9576 on the training set for ResNet50, respectively.

Similarly, as mentioned in Subsection 3.2 above, to integrate location as an additional feature, we initially integrate the geohash codes with VGG16 and ResNet50 pretrained models. We achieve top-1 accuracies of 0.9789 on the testing set and 0.9769 on the training set using the combined VGG16 and geohash feature architecture, and top-1 accuracies of 0.9812 on the testing set and 0.9795 on the training set using the combined ResNet50 and geohash feature architecture. Finally, we experimented with integrating location information into our CNN architecture by concatenating the geohash codes with the image features. By doing so, we are able to increase the top-1 accuracy on the testing set from 0.9185 to 0.9542 and on the training set from 0.9253 to 0.9512 by incorporating location as a feature.

From the results mentioned above, we can see that incorporating the geohash coding feature has led to an improvement in our classification accuracy. In each model, after integrating geohash as a location feature, there is an increase of top-1 accuracy by 2 to 3 percentage points. The reason for the small increases in the accuracies is because of the small size of the dataset. We have observed misclassifications mainly among house and building images, as they have a resemblance, but the number of misclassifications is relatively small due to our small dataset size. However, we can mitigate these misclassifications by utilizing geohash codes to differentiate between these images with similar features.

We present the results and comparisons of all the models mentioned above in Table 1. The notations used in Table 1 are as follows:

- $Acc.$ – accuracy in the testing set (general accuracy of the model);
- Acc_T – accuracy in the training set;
- $Loss$ – categorical cross-entropy loss in our multi-class classification model;

Table 1. Results and comparisons among our models based on the accuracy

Method	Acc.	Acc_T	Loss
CNN (only)	0.8842	0.9244	0.8272
CNN + Data Augmentation	0.9185	0.9253	0.4380
VGG16 (CNN)	0.9456	0.9529	0.2549
RestNet50 (CNN)	0.9516	0.9576	0.1590
CNN + Data Augmentation + Geohash Code	0.9542	0.9512	0.0954
VGG16 (CNN) + Geohash Code	0.9789	0.9769	0.0443
ResNet50 (CNN) + Geohash Code	0.9812	0.9795	0.0394

As shown in Table 1, the proposed CNN model, as well as the VGG16 and ResNet50 models, show improved accuracy after the integration of geohash codes. As shown in the table, the categorical cross-entropy loss decreases after the geohash codes have been applied, indicating that the models are able to make predictions that are closer to the true class membership probabilities. The lower loss value and similar accuracy on both the training and testing datasets suggest that the model is not overfitting to the training data.

5. CONCLUSION

This paper analyzes the limitations of using only image features in multi-class satellite image classification using CNNs. In multi-class satellite image classification, CNN architectures tend to make false predictions when there is a high degree of visual similarity between images from different classes. This issue is addressed by integrating geohash codes as an additional feature in the CNN model. With the additional geohash code feature map, the CNN model is able to make more accurate predictions.

According to the results presented in this paper, we can deduce that geohash codes can be used as an additional feature vector in the CNN architecture to make correct predictions and increase accuracy in satellite image classification. However, this may not apply in scenarios where

there is no correlation between geographical location and image content. To build a robust model for satellite image classification, it is important to take into account a range of factors, such as the size and preprocessing of the dataset, integrating additional feature vectors, the risk of overfitting, and the CNN architecture itself. This is because even a well-designed architecture may produce poor results if there is insufficient data or if the data has not been efficiently preprocessed.

In the future, we plan to explore the use of hybrid models in satellite image classification. The K-NN machine learning algorithm will be one of our focuses to identify the K-number of images that are most similar to each other based on their geohash codes, and then to automatically classify them into their respective classes. This approach has the prospect of increasing the accuracy of classifying images that are difficult to distinguish based on visual features alone, and may enable real-time classification of satellite imagery for applications such as disaster management and environmental monitoring.

ACKNOWLEDGMENT

This material is based in part upon work supported by the National Science Foundation under Grant Nos. MRI20 CNS-2018611, MRI CNS-1920182, B-BROIPS FDEP C-2104, and DHS E2055778.

REFERENCES

[1] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Transactions on Neural Network Learning Systems*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[2] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computing*, vol. 29, no. 9, pp. 2352–2449, Sep. 2017, doi: 10.1162/NECO_A_00990.

[3] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," *Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017*, vol. 2018-January, pp. 588–592, Feb. 2018, doi: 10.1109/ICCSP.2017.8286426.

[4] V. Mnih, "Machine Learning for Aerial Image Labeling," 2013.

[5] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 645–657, Feb. 2017, doi: 10.1109/TGRS.2016.2612821.

[6] F. Zhang, M. Yan, C. Hu, J. Ni, and Y. Zhou, "Integrating Coordinate Features in CNN-Based Remote Sensing Imagery Classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, 2022, doi: 10.1109/LGRS.2020.3045744.

[7] K. Tang, M. Paluri, L. Fei-Fei, R. Fergus, and L. Bourdev, "Improving Image Classification With Location Context." pp. 1008–1016, 2015.

[8] J. Wang and L. Perez, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning".

[9] R. P. de Lima and K. Marfurt, "Convolutional Neural Network for Remote-Sensing Scene Classification: Transfer Learning Analysis," *Remote Sensing 2020, Vol. 12, Page 86*, vol. 12, no. 1, p. 86, Dec. 2019, doi: 10.3390/RS12010086.

[10] C. Zhou, H. Lu, Y. Xiang, J. Wu, and F. Wang, "GeohashTile: Vector Geographic Data Display Method Based on Geohash," *ISPRS International Journal of Geo-Information 2020, Vol. 9, Page 418*, vol. 9, no. 7, p. 418, Jun. 2020, doi: 10.3390/IJGI9070418.

[11] K. Huang, G. Li, and J. Wang, "Rapid retrieval strategy for massive remote sensing metadata based on GeoHash coding," *https://doi.org/10.1080/2150704X.2018.1508907*, vol. 9, no. 11, pp. 1070–1078, Nov. 2018, doi: 10.1080/2150704X.2018.1508907.

[12] R. Moussalli, M. Srivatsa, and S. Asaad, "Fast and flexible conversion of geohash codes to and from latitude/longitude coordinates," *Proceedings - 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2015*, pp. 179–186, Jul. 2015, doi: 10.1109/FCCM.2015.18.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[14] J. Shijie, W. Ping, J. Peiyi, and H. Siping, "Research on data augmentation for image classification based on convolution neural networks," *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, vol. 2017-January, pp. 4165–4170, Dec. 2017, doi: 10.1109/CAC.2017.8243510.

[15] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard, "Adaptive data augmentation for image classification," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2016-August, pp. 3688–3692, Aug. 2016, doi: 10.1109/ICIP.2016.7533048.

[16] R. Ribani and M. Marengoni, "A Survey of Transfer Learning for Convolutional Neural Networks," *Proceedings - 32nd Conference on Graphics, Patterns and Images Tutorials, SIBGRAPI-T 2019*, pp. 47–57, Oct. 2019, doi: 10.1109/SIBGRAPI-T.2019.00010.

[17] D. Theckedath and R. R. Sedamkar, "Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks," *SN Comput Sci*, vol. 1, no. 2, pp. 1–7, Mar. 2020, doi: 10.1007/S42979-020-0114-9/TABLES/5.

[18] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, Accessed: Mar. 20, 2023. [Online]. Available: https://arxiv.org/abs/1409.1556v6

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition." pp. 770–778, 2016. Accessed: Mar. 20, 2023. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/

[20] K. Lee, R. K. Ganti, M. Srivatsa, and L. Liu, "Efficient Spatial Query Processing for Big Data," 2014, doi: 10.1145/2666310.2666481.

Arpan is a Ph.D. student at the Florida International University. He is currently working as a research assistant at the High Performance Database Research Center (HPDRC) lab. His primary research is focused on the domain of neural network and geospatial data. At present, he is doing research on advanced deep learning models like Recurrent Neural Networks (RNN) and Transformers, with the goal of applying these models to various unique domains.

Rishe [http://cake.fiu.edu/Rishe] is the inaugural outstanding university professor for Florida International University (FIU). He is a professor at FIU's Knight Foundation School of Computing and Information Sciences. Rishe is the Director of FIU's High-Performance Database Research Center. aFrom 2013 to 2022, Rishe was the executive director of a multi-

university Center for Advanced Knowledge Enablement sponsored by the National Science Foundation's industry-university cooperation program. Rishe's research has been funded by the U.S. Government and Industry at over $50 million. Rishe's inventions include 26 U.S. patents. Rishe is a Fellow medalist of the National Academy of Inventors. In 2021, Rishe received the IBM Global University Programs award. Rishe's publications include 440 papers and six books. Rishe's pioneering geospatial system TerraFly has been highlighted by the National Science Foundation in its reports to U.S. Congress.