DSE313 Assignment: Phase 2 Report

Feature Selection for few-shot building instance classification using Genetic Algorithms

Group 10

Gandhi Ananya Amit (20319)

Devashish Tripathi (21093)

Kartikey Singh (20146)

Saksham (20240)

Abstract

In the urban landscape, buildings with both a usage specified to them(i.e. single-type) and those that serve multiple purposes(i.e. mixed-type) are common. While it is easy to map single-type buildings, it proves difficult to do so for mixed types. This work proposes the use of Genetic Algorithms and Contrastive Learning to classify building usage types from Street-View images. For this phase, we continue our work from the past to train a model to get image embeddings of single-type images, from which class-wise centroids are calculated, which are then used to find the percentage composition of the mixed-type buildings. Triplet Loss was used for training purposes, and works like this, with a few more modifications, can be used for labelling mixed-type buildings to the class they have the maximum percentage composition of.

1. Introduction

In developing countries like India, it is common to see a single building housing multiple ventures. The most common example of this would be buildings that have a grocery store, a commercial building type at the lower floors, and a residential part above these stores. Such buildings can be found all over India in various localities. Other similar examples include abandoned buildings that have survived several decades and are now being reused for commercial or residential purposes on the lower floors. Such buildings are common in the 'older' parts of Indian cities. Industrial buildings may have parts of them under construction. These are just a few examples of the different types of combinations that can be present in mixed-type buildings.

For this task, we have assumed that there are seven broad classes a single-type building can belong to: Abandoned, Commercial, Industrial, Religious, Residential, Under-Construction and Others. A mixed-type building can belong to a combination of any of these categories. As such, marking the mixed-type buildings is arduous as it may not be immediately evident which category such buildings best fit into. As such, we decided to instead classify the category-wise percentage composition of mixed-type buildings from their *Street-View* images.

For this purpose, the task was divided into two phases. Phase 1 involved the creation of a dataset of mixed-type buildings and selecting a base model for the task of classifying single-type buildings into the seven categories defined earlier. This base model was then tuned for the task of getting the percentage composition of mixed-type buildings in the second phase. This was done through the use of, among many techniques, Genetic Algorithms and Triplet Loss.

1.1. Brief theoretical overview

One important work done for building classification [2], which used *Street-View* images to classify the use type of single-type buildings and marked the building usage on an aerial view image of the building's locality. For this task, *Street-View* images were taken, outliers were removed using a pre-trained CNN on the Places2 dataset, and four models, ResNet18, ResNet34, AlexNet and VGG16, were experimented with and fine-tuned for the classification task. VGG16 was finally selected based on the best F1 scores and accuracy.

Contrastive Learning is a Self-Supervised Learning paradigm used to combat the requirement for a high amount of labelled data for training CNNs. This method works similarly to kNN clustering in the sense that the data points that match are clustered closer while the negative matches are distanced. It allows the model to cluster the data points based on the high-level features and, thus, reduces the need for labelled data.

Triplet Loss, first described in [6], was introduced for the task of Face recognition and clustering. It works by taking three inputs: an anchor point, a positive point and a negative point. The anchor and the positive point are more similar than the anchor and the negative point. As such, this loss aims to minimize the 'distance' between the anchor and the positive point and maximize the distance between the anchor and the negative point. As such, it works in a way to implement Contrastive Learning.

By using Triplet Loss, we intend to reduce the 'distance' between similar images and increase the distance between dissimilar images so as to get better clusters of each class, which will be used for the category-wise percentage calculation. Mathematically,

$$L(A, P, N) = \max(0, d(A, P) - d(A, N))$$

where L is the triplet loss, A is the anchor point, P is the positive point, N is the negative point, and d is the distance function.

Genetic Algorithms work similarly to natural selection, where parts of two genes are merged, a mutation may occur, and out of a population, the strongest genes are allowed to mate, leading to a new population. We have decided to use Genetic Algorithms to select the best weights for which we are getting results.

We have combined Triplet Loss and Genetic Algorithms to get a model that would give us image embeddings of the single-type images for all the classes, from which we plot the centroids of each class, using PCA for dimensionality reduction. This is followed by the inference of the mixed-type images, where the images are passed through the model, and their embedding is found, which is reduced to 2 dimensions using PCA and plotted on the graph. The distances of the image's point from each centroid are calculated, which is used for calculating the percentage composition using normalisation.

$$PercentComp_c = d(c,x)/\sum(d(c_j,x))$$

where c_j is the jth class' centroid, x is the representation of the mixed-type image, and d is Euclidean distance.

1.2. Summary of Phase 1

The *Street View* images of mixed-type buildings were located and downloaded using the software *Street View Download 360*. These images were converted to a FOV form using the *equirectangular-toolkit*, which can be found at [3]. A dataset of 100 mixed-type images was created, which can be found here.

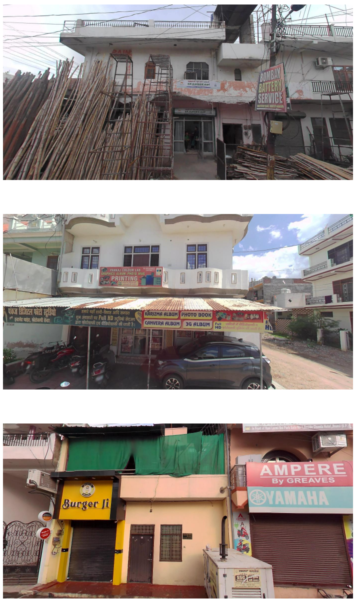


Figure 1. *Sample mixed-type images*

After the dataset creation, two pre-trained models, ResNet-50[7] and DINOv2 [5], were used through transfer learning to train a classifier to classify a single-type building image into one of the seven categories defined earlier. The models were evaluated on criteria such as Accuracy, Confusion Matrices and Classification Reports, and DINOv2 was selected as the best model based on these criteria. A possible reason for DINOv2 outperforming ResNet-50 discussed in the Phase-1 report was considered to be the 'Attention' mechanism present in the DINOv2 model, which allowed it to have more detailed feature representations of the image. Adding onto that, ResNet-50 is a CNN-based model, while DINOv2 is based on pre-trained visual models, trained with different Vision Transformers (ViTs) [5]. As discussed in [4], ViTs are significantly less biased towards local textures compared to CNNs. As such, CNNs, which have access to only a small receptive field, have to work with less information, while ViTs, thanks to attention layers, have a sufficiently sized receptive field to capture all the relevant information. As such, we selected DINOv2 as the base model for the calculation of the category-wise percentage composition in the second phase.

The implementation details are mentioned in the 'Methodology' section. The results obtained and discussions on them are in the 'Results' and the 'Discussions' sections, respectively.

2. Methodology

The following steps were taken to implement the phase two work.

- **Creation of Triplets:** Triplets of images were created. For each anchor image, a random image from the same category which was not the anchor image, was taken as the positive image, and a random image from another class was taken as the negative image. As such, we ended up with a total of 3772 triplets.

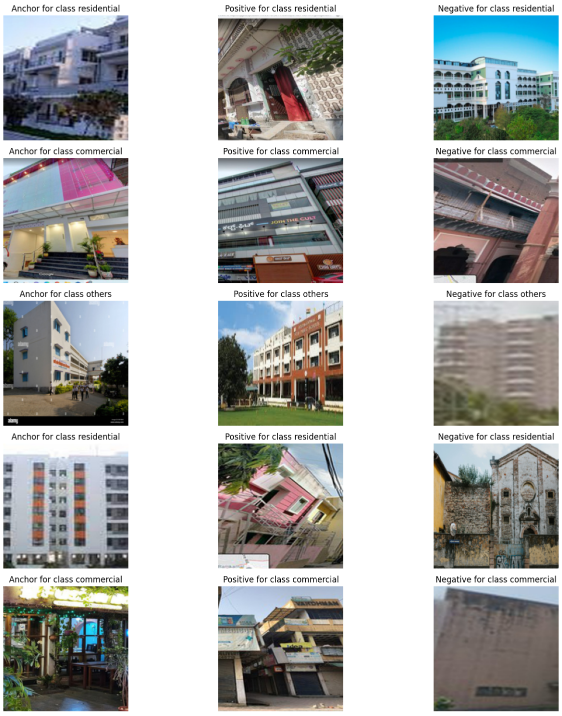

Anchor for class residential
Positive for class residential
Negative for class residential
Anchor for class commercial
Positive for class commercial
Negative for class commercial
Anchor for class others
Positive for class others
Negative for class others
Anchor for class residential
Positive for class residential
Negative for class residential
Anchor for class commercial
Positive for class commercial
Negative for class commercial

Figure 2. *5 randomly selected triplets*

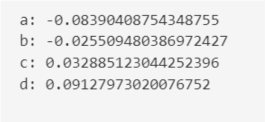
After the creation of triplets, a triplet loss function was implemented.

- **Neural Network implementation:** For the purpose of getting the weights to implement for Genetic Algorithm, we implemented neural networks add-ons to the base DINOv2 model with 0, 1 and 2 hidden layers, which were each trained for 8 epochs and on different activation functions. For this step, various sizes of the hidden layers were tried, such as 256, 128, 512 and 64. Activation functions used included ReLU, LeakyReLu, tanh and GELU. Dropout was also used, with values such as 0.05, 0.15 and 0.25. The best model that was selected was a model with 2 hidden layers, with the neuron structure as:

$$1536- > 256- > 128- > 64$$

with GELU activation function, a dropout of 0.25 for both the layers and trained for 8 epochs. The batch size used was 32, with a validation split of 0.25.

- **Selecting weights for application of Genetic Algorithm:** Once the model was trained, its weights were downloaded and copied into a final model structure. Histograms of the neural network weights were plotted. The weights of the final classifier layer were reduced to 4 bins. The centroids of these 4 bins were then selected as the 4 genes. The gene values are in the Figure 3



```
a: -0.08390408754348755
b: -0.025509480386972427
c: 0.032885123044252396
d: 0.09127973020076752
```

Figure 3. *Gene values*

The various weight histograms are shown in Figure 4.

- **Applying Genetic Algorithm:** For the implementation of the Genetic Algorithm, the PyGAD library [1] was used. The final layer of the neural network added at the end of the DINOv2 model had (128)*64= 8192 neurons. As such, taking the 4 genes from above, we created a **population of size 6** generated randomly, where each member has **8192 genes**. These weights then replace the neural network of the final model, and we run inference on the validation set only in order to select the 2 best children. The loss function is Triplet Loss, and we aim to minimize it. The fitness function is the inverse of the Triplet Loss. We evaluate only the validation set as we suffice with only having the 2 best-performing weight arrays for children as the parents for the next generation, which the validation set is enough to indicate; hence, there is no need to run the whole triplet loss model on the whole of the training data. After taking the top 2 members of the population as the parents for the next generation, 6 offspring are produced. This was done for 3 iterations, and the best offspring of the final iteration was selected as the weights array for the final model.

- **Getting inference for the mixed-type buildings:** Using this final model, the single-type images were sent through it, and image representations were received per class. These representations were concatenated per class, and PCA was used to reduce the representation to two dimensions. The centroids per class were calculated and plotted on a scatter plot. Finally, the mixed-type images were sent through the model, and their representations were obtained, which were reduced to two dimensions and plotted on the scatter plot. Then, the distances between different centroids and the point were calculated, and the values were normalized. Thus, the percentage composition per class was received.

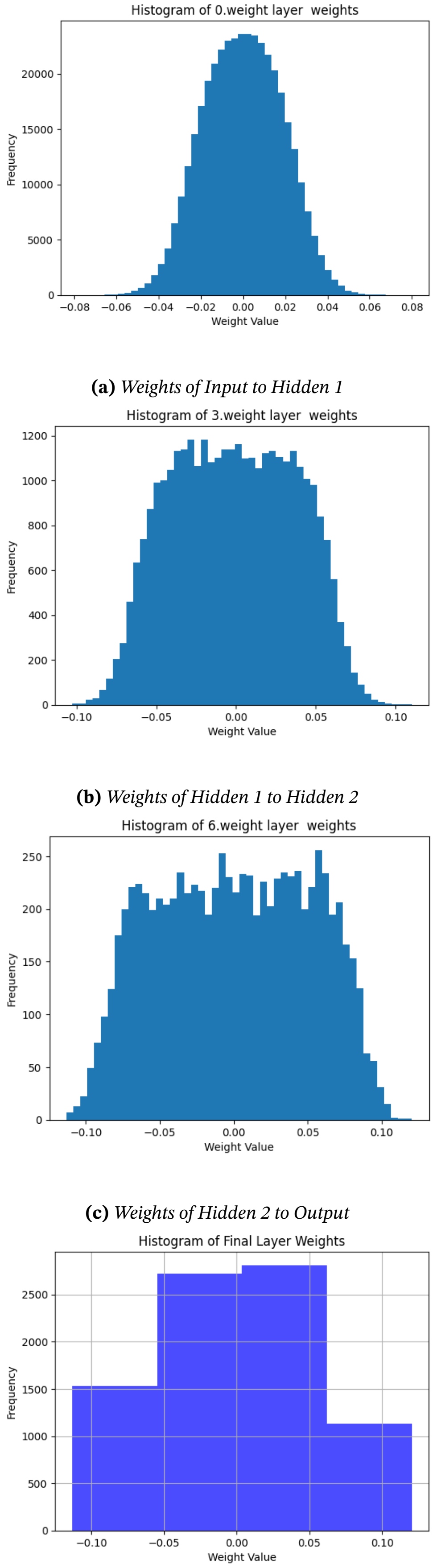
3. Results

The loss curves for the best neural network are in Figure 5.

The scatter plots for the centroids of the seven classes using the image embeddings received from the final model using PCA are in Figure 6.

Sample percentage composition for the first few images of the mixed-type buildings are in Figure 7.

The first 10 and the Last 10 weights received by the Genetic Algorithm are in Figure 8.



(a) *Weights of Input to Hidden 1*

(b) *Weights of Hidden 1 to Hidden 2*

(c) *Weights of Hidden 2 to Output*

(d) *The final layer weights in 4 bins*

Figure 4. *Weight Histograms*

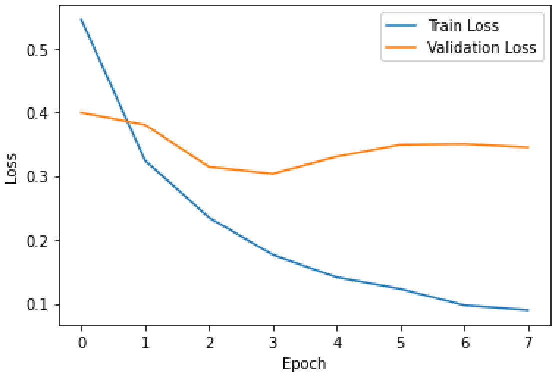


Figure 5. *Train and Validation Loss Curves*

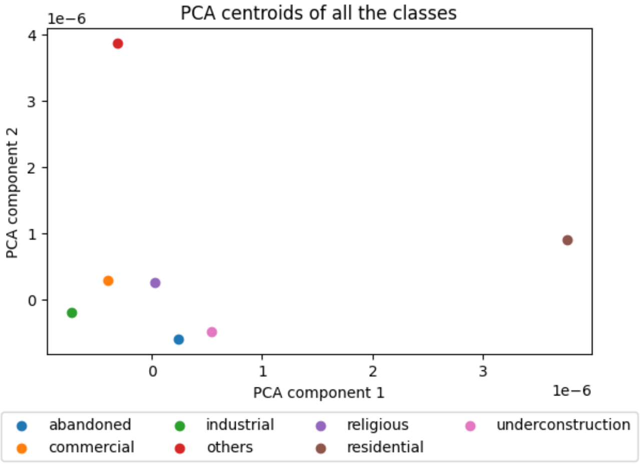
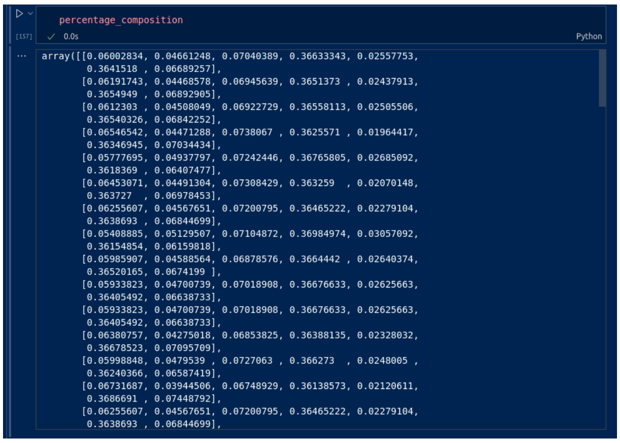


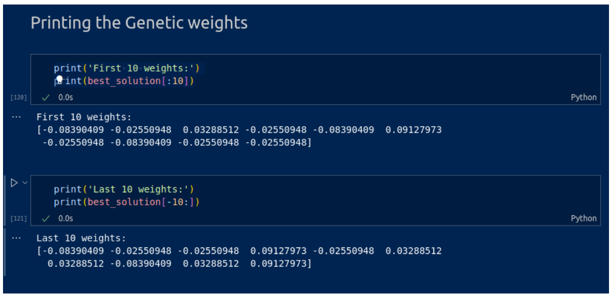
Figure 6. *Centroids of the seven classes*



```
percentage_composition
[157]  ✓ 0.0s                                                        Python

array([[0.06002834, 0.04661248, 0.07040389, 0.36633343, 0.02557753,
        0.3641518 , 0.06689257],
       [0.06191743, 0.04468578, 0.06945639, 0.3651373 , 0.02437913,
        0.3654949 , 0.06892905],
       [0.0612303 , 0.04508049, 0.06922729, 0.36558113, 0.02505506,
        0.36540326, 0.06842252],
       [0.06546542, 0.04471288, 0.0738067 , 0.3625571 , 0.01964417,
        0.36346945, 0.07034434],
       [0.05777695, 0.04937797, 0.07242446, 0.36765805, 0.02685092,
        0.3618369 , 0.06407477],
       [0.06453071, 0.04491304, 0.07308429, 0.363259  , 0.02070148,
        0.363727  , 0.06978453],
       [0.06255607, 0.04567651, 0.07200795, 0.36465222, 0.02279104,
        0.3638693 , 0.06844699],
       [0.05408885, 0.05129507, 0.07104872, 0.36984974, 0.03057092,
        0.36154854, 0.06159818],
       [0.05985907, 0.04588564, 0.06878576, 0.3664442 , 0.02640374,
        0.36520165, 0.0674199 ],
       [0.05933823, 0.04700739, 0.07018908, 0.36676633, 0.02625663,
        0.36405492, 0.06638733],
       [0.05933823, 0.04700739, 0.07018908, 0.36676633, 0.02625663,
        0.36405492, 0.06638733],
       [0.06380757, 0.04275018, 0.06853825, 0.36388135, 0.02328032,
        0.36678523, 0.07095709],
       [0.05998848, 0.0479539 , 0.0727063 , 0.366273  , 0.0248005 ,
        0.36240366, 0.06587419],
       [0.06731687, 0.03944506, 0.06748929, 0.36138573, 0.02120611,
        0.3686691 , 0.07448792],
       [0.06255607, 0.04567651, 0.07200795, 0.36465222, 0.02279104,
        0.3638693 , 0.06844699],
```

Figure 7. *Percentage composition*



Printing the Genetic weights

```
print('First 10 weights:')
print(best_solution[:10])
[120]  ✓ 0.0s                                                        Python

First 10 weights:
[-0.08390409 -0.02550948  0.03288512 -0.02550948 -0.08390409  0.09127973
 -0.02550948 -0.08390409 -0.02550948 -0.02550948]
```

```
print('Last 10 weights:')
print(best_solution[-10:])
[121]  ✓ 0.0s                                                        Python

Last 10 weights:
[-0.08390409 -0.02550948 -0.02550948  0.09127973 -0.02550948  0.03288512
  0.03288512 -0.08390409  0.03288512  0.09127973]
```

Figure 8. *First and Last 10 weights*

The mixed-type buildings(in red) representations in 2d plotted on the centroids plot are in Figure 9.

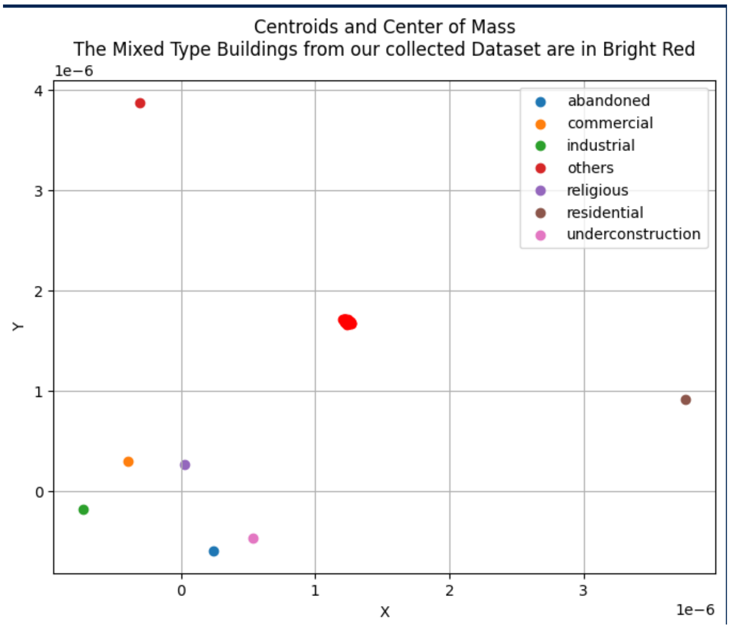


Figure 9. *Mixed types on Centroids plot*

4. Discussions

Using triplet loss, we ensured the model knew how to cluster images based on the representations. This is evident in Figure 6, which shows the centroids of the seven classes. It becomes increasingly obvious that the images of the residential class are way different than the other classes. It's quite apparent that the residential class stands out from the rest, indicating that the model has successfully learned to recognize the unique characteristics of residential buildings.

When plotting the representations of the mixed-type images as two-dimensional PCA to the centroid plots, we discovered that all mixed-type buildings do come in the centre of the plot because they belong to the mixed class and, as such, consist of properties of all the classes to some extent.

However, from the percentage compositions received from our genetic algorithm implementation and the inference method, we inferred that most buildings were coming as belonging to the class 'others' or 'residential'. This could have two reasons. The primary reason is that the single-type dataset consists mostly of images belonging to the 'others' class, followed by the images belonging to the residential class, making the model representations more biased towards these classes. Secondly, the images in the mixed-type dataset were mostly taken in a residential setting. As such, most buildings were residential, with another component, which the model classified as others. This might be due to the single-type dataset again.

Our application of Genetic Algorithm could have been made more effective by running the algorithm for more number of iterations and by increasing the number of generations. This could have helped in getting even more clearer weights and percentage compositions. This is because The performance of the Genetic Algorithm depends on several factors, like the number of solutions (population size), the number of generations, and the way the solutions are combined and mutated.

To enhance the results and broaden the applicability of this approach, future research could explore several directions. For example, trying out different model architectures, loss functions, and optimization algorithms could potentially improve the learning of image features and the overall classification performance. Additionally, increasing the size and diversity of the dataset by including more building types and locations would make the model more robust and capable of generalizing to various scenarios.

5. Conclusion

In this work, we proposed a method for classifying building usage types from Street-View images, which we had taken from the Google Maps Street View Image in Phase 1. Using Genetic Algorithms and Contrastive Learning, our goal was to train a model to obtain image embeddings of single-type images, calculate class-wise centroids, and use them to determine the percentage composition of mixed-type buildings.

Our methodology involved creating triplets of images, implementing neural networks with various configurations (different numbers of Hidden Layers), selecting weights(genes) for the application of the Genetic Algorithm based on Histogram binning, mixing and matching to get the best weights array and obtaining inferences for mixed-type buildings. Triplet Loss was used as the loss function in the Training Part of the Neural Network Addons and the Genetic Algorithm.

In the results section, we demonstrated that the model was able to cluster images effectively, as evident from the scatter plot of the centroids of the seven classes. The distribution of the centroids in Figure 5 asserts that our approach is sound, as the location of the centroids makes logical sense. We were also able to use the Genetic Algorithm to obtain the percentage composition of the mixed-type buildings. Additionally, we were able to use the PyGAD library effectively, showing its robustness for Genetic Algorithm implementation. Better results could have been obtained if Genetic Algorithm was implemented more robustly.

Finally, the results showed how the need for a dataset of single-type buildings made on more classes and of higher size, as a wider dataset based on images of Indian buildings would allow better performance. Our model suffered from the mixed-type images being taken mostly in a residential context, and the single-type buildings upon whose triplets it was trained mostly belonged to the 'others' or the 'residential' class. Thus, focusing on the collection of properly labelled buildings images taken in a wider range of locations can be done.

The proposed approach has the potential to be extended to other domains where mixed-type classification is required. Future work could focus on refining the Genetic Algorithm implementation, exploring alternative architectures, and expanding the dataset to include a wider range of building types and locations.

In conclusion, this work presents a novel approach to classifying building usage types, especially Mixed Indian Buildings, which are harder to classify into just a particular Data Type from Street-View images using Genetic Algorithms and Contrastive Learning. How this study was done and what it found can be a good starting point for more research on this topic. It also shows how powerful it can be to use deep learning together with optimization techniques and meta-heuristic approaches to solve difficult classification problems.

References

[1] Ahmed Fawzy Gad. "Pygad: An intuitive genetic algorithm python library". In: *Multimedia Tools and Applications* (2023), pp. 1–14.

[2] Jian Kang et al. "Building instance classification using street view images". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 145 (2018). Deep Learning RS Data, pp. 44–59. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2018.02.006. URL: https://www.sciencedirect.com/science/article/pii/S0924271618300352.

[3] Nitish Mutha. *GitHub - NitishMutha/equirectangular-toolbox: Handy tool for equirectangular images — github.com*. https://github.com/NitishMutha/equirectangular-toolbox.

[4] Muzammal Naseer et al. "Intriguing Properties of Vision Transformers". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=o2mbl-Hmfgd.

[5] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2023. arXiv: 2304.07193 [cs.CV].

[6] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015. DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[7] Ross Wightman, Hugo Touvron, and Herve Jegou. "ResNet strikes back: An improved training procedure in timm". In: *NeurIPS 2021 Workshop on ImageNet: Past, Present, and Future*.

A. Contribution:

The contributions of each individual are as follows:

Ananya: Application of the Genetic Algorithm to find the best weights array, implemented the binning of the weights of the neural networks, training the final model based on the Genetic Algorithm and wrote sections 2(Genetic Algorithm Part), 3, 4 and 5 of the report.
Devashish: Implemented the creation of the dataset in triplets form, trained the various neural networks, did inference on the mixed-type images and wrote the abstract, sections 1 and 2 of the report.
Kartikey: Made Triple Loss Function and Addon Hidden Layers functions.
Saksham: Searched for references to put in the paper.