ENEL 525 Fall 2024 - Final Project
Arial Image Classification with Convolutional Neural Networks (CNNs)

ENEL 525: Machine Learning for Engineers
University of Calgary
Submitted by: Tania Rizwan (30115533)
Date Submitted: December 18, 2024

1 Introduction

The objective of this project is to implement deep learning techniques to classify high-resolution aerial images into 21 land-use categories using the UC Merced Land Use Dataset. This dataset contains 2,100 images collected from aerial and satellite data, each labeled into one of 21 distinct classes such as agricultural, residential, forest and recreational areas. Each class contains exactly 100 RGB images with a resolution of 256x256 pixels, manually extracted from the USGS National Map Urban Area Imagery collection, making the dataset balanced [1][2]. A snippet of the dataset is shown in Figure 1.

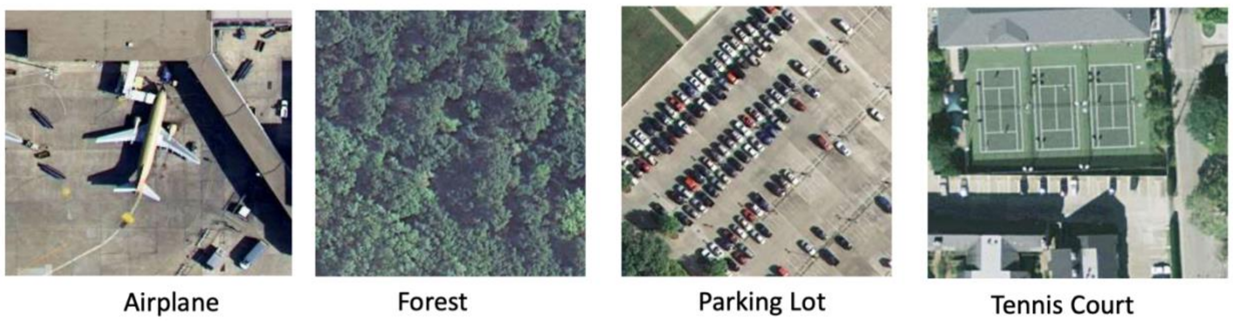


Figure 1. Snippet of the UC Merced Land Use Dataset [1]

To accomplish this task, convolutional neural networks (CNNs) are utilized as the primary deep learning architecture, as they excel at image classification tasks by capturing hierarchical patterns in image data. They extract features such as edges, textures and complex layers through convolutional layers, and utilize pooling to reduce dimensionality while storing the most important features in the data. This makes them practical for detecting objects in images, while making them effective at understanding the spatial and structural details in high-resolution images.

Therefore, this project aims to recognize and classify various remote sensing/aerial image categories in the dataset through exploring feature extraction using CNNs, to create a model that achieves high accuracy in distinguishing between similar land-use categories.

2 Methodology

2.1 Problem Setup

The task in this project is to build a deep learning model with high accuracy to classify 2,100 high-resolution aerial images from the UC Merced Land Use Dataset into one of 21-land use categories (e.g. agriculture, overpass, freeway, etc.). Each class contains exactly 100 images in RGB format with a resolution of 256x256 pixels. The entire dataset was used and split into training, validation and testing subsets with an additional augmentation layer to improve the model's performance.

The approach taken to accomplishing the task involved implementing a CNN using transfer learning with the InceptionResNetV2 architecture.

2.2 Network Design

The model was designed to leverage InceptionResNetV2 for feature extraction while allowing customization for the classification task. InceptionResNetV2 is a CNN consisting of 164 layers, that has been developed on more than one million images from the ImageNet database [3]. After trying a basic CNN and ResNet50, further research was done to determine which transfer learning method would be better suited to the task. InceptionResNetV2 was selected as it is one of the best models to use for image classification [4], and was the most effective model in applications such as successfully identifying individual dog breeds in a series of high resolution images, and detection of garbage from images in an experiment with various other models [5], [6].

2.2.1 Preprocessing and Data Augmentation

The data was preprocessed through a variety of steps to ensure it was properly formatted prior to being input into the neural network. The first step consisted of ensuring the images had the correct format by resizing them to have dimensions of 256x256 pixels and converting them to the RGB format. The second step involved normalizing the images to have pixel values ranging from 0 to 1. Finally, the labels array was converted to have a one-hot format (where each class is represented as one of *k* distinct options), as this is format is useful for multi-class problems.

Data augmentation was performed to generate new training samples by applying random transformations to the existing images, such as rotation, flipping, zooming and shifting. It is a common practice that is used to enhance a dataset and reduce overfitting as it increases diversity in the training data input to the model. More specifically, the following transformations were applied:

- Horizontal flipping
- Rotation by up to 20%
- Zoom by up to 20%
- Contrast adjustment by 10%
- Translation (horizontal and vertical) by 10%

2.2.2 Base Model

Transfer learning using InceptionResNetV2 was used as the base model for the CNN, with the convolution layers frozen and the weights obtained from the ImageNet dataset. Its fully connected (dense) layers were not utilized as they were more suited to the 1000 class differentiation task, whereas the UC Merced Land Use Dataset only consisted of 21 classes. Therefore, customization of the dense layers was necessary and separated from the InceptionResNetV2 model. The convolution layers of the model were also frozen to ensure they did not update during training, as this model had already been trained on an extensive set of data, therefore it was deemed capable for the task at hand.

2.2.3 Custom Layers

The final model incorporated the base model and data augmentation, along with a global average pooling, dense, dropout and a final output layer.

The global average pooling layer was used instead of the typical flattening layer as it effectively reduces spatial dimensions while keeping the depth (features) of the data. This is performed by calculating the average of each feature map that is output from the convolutional layers making it better suited for applications in CNNs. The dense layer consisted of 512 neurons as it was deemed suitable for the size of this dataset, and the ReLu activation function was added as it is a fundamental component of CNNs and is used to introduce non-linearity to enable the network to effectively learn complex patterns in the data. The dropout layer, with a 50% rate, was incorporated as a good practice to reduce overfitting by randomly deactivating half of the neurons during training. Finally, the output layer was another fully connected dense layer with 21 neurons (one per each class) with the SoftMax activation function used, as it converts the outputs into probabilities and is suitable for multi-class classification tasks. The network diagram is shown in Figure 2 (with the InceptionResNetV2 structure referenced from a research paper found online [7]).

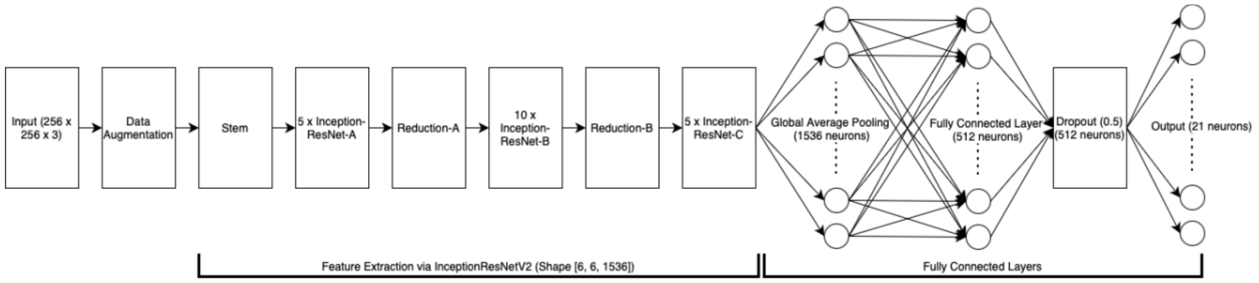


Figure 2. Network Diagram

2.3 Training Scheme

The entire dataset was split into different sections for use in the model. As per the Project Overview, 70% of the data was used for training, with 15% set aside for validation to tune the model's hyperparameters during training. The model was then compiled with the Adam optimizer, with 20 epochs (chosen through trial and error to balance performance and computation), and a batch size of 16 to ensure stability during training and to speed the process of updating the weights to help the model converge faster.

Moreover, TensorFlow's built-in callbacks were also used to reduce computation time and help the model converge faster if it was not learning effectively. The first callback involved Early Stopping and was used to stop training if the validation loss didn't improve for 5 epochs. The second callback was ReduceLRonPlateau, which aimed to reduce the learning rate by 50% if the validation loss failed to improve (plateaued) for 3 epochs.

2.4 Testing Scheme

For testing, the remaining 15% of data was set aside for evaluation. Data from external sources was not utilized for testing, as data augmentation during training was implemented to improve the model's ability to generalize to new data.

2.5 Hyperparameters and Compilation

2.5.1 Optimizer and Learning Rate

The model was compiled using the Adam optimizer, a widely used choice for classification tasks due to its computational efficiency and adaptability [8]. A learning rate of $1 \ x \ 10^{-4}$ was selected as smaller step sizes promote stable and gradual convergence during training and are suitable for CNN architectures [8].

2.5.2 Loss Function

The Categorical Cross-entropy loss function was used as it is suited for multi-classification tasks, especially with neural networks that use the SoftMax activation function in the output layer [9]. This loss function measures the difference between the predicted class probabilities, obtained from the SoftMax output, and the true one-hot encoded labels, helping the model learn to assign higher probabilities to the correct class.

The derivation of the loss function is described in the subsequent steps:

1. For a classification problem with C possible classes, we suppose that for a single input $\vec{x}$, the model outputs a probability distribution over the classes:

$$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_C \ with \sum_{k=1}^{C} \hat{y}_k = 1$$

   Here, $\hat{y}_k$ is the predicted probability that the correct answer is class k. For example, $\hat{y}_3 = 0.7$ indicates that the input $\vec{x}$ has a 70% chance of belonging to class 3.

   We also know the true label, which is a vector represented with one-hot encoding:

$$y_1, y_2, \ldots, y_C$$

   Where only one of these y_k equals 1 (the correct class) whereas the rest are set to 0.

2. The goal of the Categorical Cross-Entropy Loss function is to make the predicted probabilities match the true labels as closely as possible and therefore to maximize the probability that the model assigns the input to the correct class [10]. This can be done by

utilizing the principle of maximum likelihood [7]. We can say that if the correct class is c, then the probability the model assigns to it is $\hat{y}_c$, and we want to maximize the following:

$$P(correct\ class, c) = \hat{y}_c$$

The likelihood is represented as the product of the predicted probabilities for all training samples, which we want to maximize.

3. We then take the logarithm to simplify the calculation, which turns these products into a sum and can be represented as follows:

   $$\log(\hat{y}_c)\ for\ one\ training\ sample, and \sum_{i=1}^{N} \log\,(\hat{y}_{c_i})\ for\ all\ training\ samples$$

   Here, $\hat{y}_{c_i}$ is the probability the model gives to the correct class for the i-th training sample, where the sample consists of the input vector along with its target output.

4. We can then maximize $\sum \log\,(\hat{y}_{c_i})$ which is the same as minimizing its negative. We can assign a negative sign to turn the problem into a typical optimization problem:

   $$-\sum_{i=1}^{N} \log\,(\hat{y}_{c_i})$$

5. Since we know the true label vector $\vec{y}$ is encoded in the one-hot format, we can write $\log(\hat{y}_c)$ as:

   $$\log(\hat{y}_c) = \sum_{k=1}^{C} y_k \log\,(\hat{y}_k)$$

   because only the correct class c has $y_c = 1$ whereas all the remaining classes have $y_k =$
0.

6. Combining everything together, the loss finally becomes:

   $$Loss = -\sum_{i=1}^{N}\sum_{k=1}^{C} y_k^i \log\,(\widehat{y_k^i})$$

Then we can take the average over all N training samples to achieve the final Categorical Cross-Entropy Loss function:

$$Categorical\ Cross - Entropy\ Loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} y_k^i \log\,(\widehat{y_k^i})$$

Note: The following references were used for this derivation: [7], [10], [11], [12], [13],[14].

2.5.3 Metrics

Accuracy, F1 score, Precision and Recall were used to evaluate the model's performance as these are commonly used for classification tasks. The F1 score indicates how well the model handles both false positives and negatives. Precision measures how many predicted positives are correct, while recall measures the model's ability to identify all true positives.

2.6 Backpropagation in CNNs

Backpropagation is a method used to improve a neural network's performance by updating its weights through feeding the output back into the model. It works by determining how much each weight contributes to the error (loss) and performs adjustments to reduce that error in future predictions.

There are three key steps involved in backpropagation. The first step is called the Forward Pass, which involves allowing the input data to flow through the network, layer by layer, to produce a prediction. The next step involves the Loss Calculation, as the network's predictions are compared to the actual labels using a loss function, which gives the error (loss). This is then sent backward through the network using calculus (derivatives and the chain rule) to determine how much each weight and filter contributed to the error, which are then updated to reduce the error and improve the model's predictive capabilities.

In CNNs, backpropagation is used to fine-tune the filters in the convolutional layers, and the weights in the fully connected layers, as these are the key components that detect and classify patterns in data. By adjusting these parameters layer by layer, backpropagation enables the network to improve its ability to detect key features such as edges, shapes and textures in each image that is input into the model, and perform better at the overall task [15].

3 Results and Discussion

This section presents the performance of the CNN model using InceptionResNetV2 as the backbone for classifying images from the UC Merced Land Use Dataset.

3.1 Training and Validation Performance

The training accuracy/loss and validation accuracy/loss curves are shown in Figure 3 below.

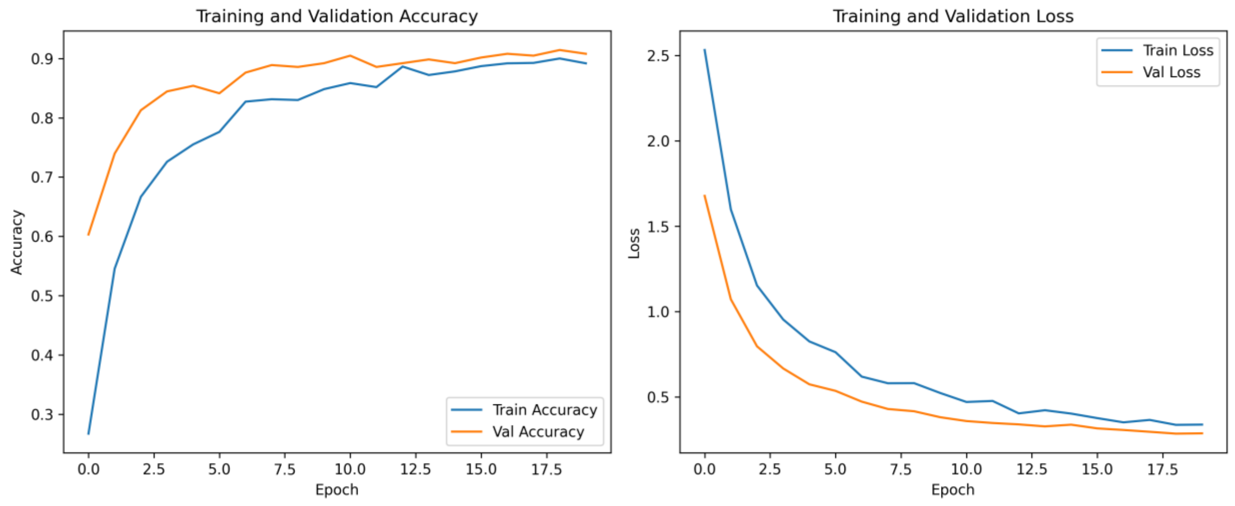


Figure 3. Training and Validation Accuracy and Loss Curves vs Number of Epochs

The Training and Validation Accuracy (left plot) shows that both accuracies increased smoothly across epochs, finally stabilizing around 90% during the last few epochs. The Training and Validation Loss (right plot) shows the loss decreasing steadily for both sets, finally stabilizing around 0.30. Both plots indicate that the model converged effectively without overfitting or underfitting.

3.2 Model Summary and Training Parameters

Furthermore, relevant training parameters are summarized in Table 1. The model utilized the 164-layer deep InceptionResNetV2 as the backbone for feature extraction, along with a global average pooling layer to replace flattening to reduce dimensions, followed by a 512-neuron dense layer with the ReLu activation function to introduce non-linearity and help the CNN extract complex features, and a 50% dropout layer for regularization. The final SoftMax layer classified the input into 21 classes and the model was trained using the Adam optimizer with a small learning rate to ensure stable convergence. A batch size of 16 samples was used along with categorical cross-entropy loss over 20 epochs. Early stopping was used to prevent unnecessary computations, and ReduceLRonPlateau adjusted the learning rate dynamically to improve performance.

Table 1. Summary of Training Parameters

Parameter	Value
Learning Rate	$1 x 10^{-4}$
Optimizer	Adam
Loss Function	Categorical Cross-Entropy
Batch Size	16 samples
Epochs	20
Callbacks	Early Stopping, ReduceLRonPlateau
Base Model	InceptionResNetV2
Dropout Rate	50%
Metrics	Accuracy, F1 score, Precision and Recall

3.3 Overall Performance

The results for training, validation and testing are shown in

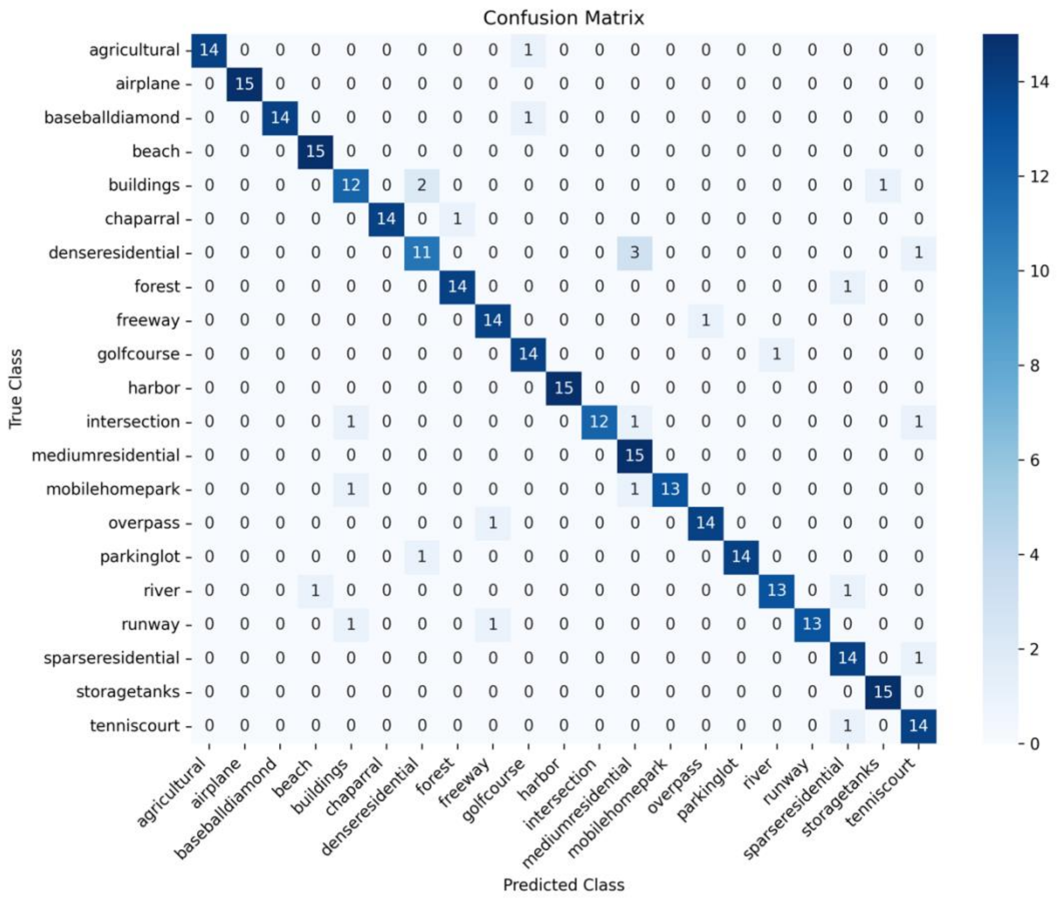
Table 2.

Table 2. Overall Performance

	Performance		
Metric	**Training**	**Validation**	**Testing**
Accuracy	0.89	0.91	0.92
F1 Score (averaged across all classes)	0.88	0.91	0.92
Loss	0.34	0.29	0.29
Precision	0.93	0.94	0.95
Recall	0.85	0.88	0.89

The model demonstrated strong and consistent performance across the training, validation and testing datasets, indicating that it learned and generalized effectively. This is especially demonstrated by the improvement in performance metrics from training to testing. More particularly, the accuracy improved from 0.89 to 0.92, reflecting the model's ability to generalize well to unseen data without overfitting (as overfitting would be indicated by an incredibly high accuracy), which could be credited to the use of data augmentation during training. Similarly, the F1 score closely matched the accuracy values, confirming the model's effectiveness in managing both false positives and false negatives consistently. Moreover, the loss steadily decreased from 0.34 during training to 0.29 during testing, which highlights that the model successfully converged. High precision indicates that the model accurately minimized false positives, while improved recall (from 0.85 during training to 0.89 during testing) demonstrates the model's ability to correctly identify true positives.

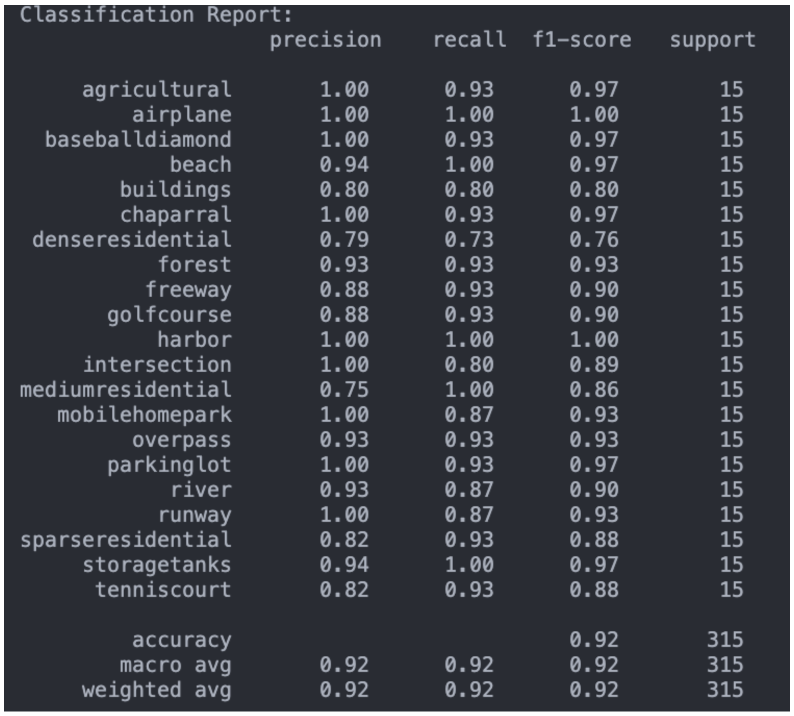
These results suggest that the model is robust and well-suited for the multi-class classification task and has a strong potential in performing effectively on similar, high resolution image datasets.



True Class \ Predicted Class	agricultural	airplane	baseballdiamond	beach	buildings	chaparral	denseresidential	forest	freeway	golfcourse	harbor	intersection	mediumresidential	mobilehomepark	overpass	parkinglot	river	runway	sparseresidential	storagetanks	tenniscourt
agricultural	14	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
airplane	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
baseballdiamond	0	0	14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
beach	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
buildings	0	0	0	0	12	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0
chaparral	0	0	0	0	0	14	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
denseresidential	0	0	0	0	0	0	11	0	0	0	0	0	3	0	0	0	0	0	0	0	1
forest	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	1	0	0
freeway	0	0	0	0	0	0	0	0	14	0	0	0	0	0	1	0	0	0	0	0	0
golfcourse	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	1	0	0	0	0
harbor	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0
intersection	0	0	0	0	1	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	1
mediumresidential	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0
mobilehomepark	0	0	0	0	1	0	0	0	0	0	0	0	1	13	0	0	0	0	0	0	0
overpass	0	0	0	0	0	0	0	0	1	0	0	0	0	0	14	0	0	0	0	0	0
parkinglot	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	14	0	0	0	0	0
river	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	13	0	1	0	0
runway	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	13	0	0	0
sparseresidential	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	1
storagetanks	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0
tenniscourt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	14

Figure 4. Confusion Matrix

The confusion matrix in Figure 4 shows the number of both correctly and incorrectly classified samples across all 21 classes in the testing dataset (which consisted of 15 images per class, for a total of 315 images). Most classes (such as Airplane, Beach, Storage Tanks, etc.), achieved perfect or near-perfect accuracy, with all 15 samples correctly identified, as shown across the diagonal. However, a few classes had errors indicating visual similarities, such as Dense Residential for which most errors involved misclassification as Medium Residential. Some images from the Buildings class were also misclassified as Dense Residential, again likely due to similarity. Other minor misclassification errors occurred for classes such as River and Sparse Residential, however the number of errors is significantly low as compared to the set of images classified correctly, indicating that the model performed well overall.



```
Classification Report:
                    precision    recall  f1-score   support

      agricultural       1.00      0.93      0.97        15
          airplane       1.00      1.00      1.00        15
   baseballdiamond       1.00      0.93      0.97        15
             beach       0.94      1.00      0.97        15
         buildings       0.80      0.80      0.80        15
         chaparral       1.00      0.93      0.97        15
  denseresidential       0.79      0.73      0.76        15
            forest       0.93      0.93      0.93        15
           freeway       0.88      0.93      0.90        15
        golfcourse       0.88      0.93      0.90        15
            harbor       1.00      1.00      1.00        15
      intersection       1.00      0.80      0.89        15
 mediumresidential       0.75      1.00      0.86        15
    mobilehomepark       1.00      0.87      0.93        15
          overpass       0.93      0.93      0.93        15
        parkinglot       1.00      0.93      0.97        15
             river       0.93      0.87      0.90        15
            runway       1.00      0.87      0.93        15
 sparseresidential       0.82      0.93      0.88        15
      storagetanks       0.94      1.00      0.97        15
       tenniscourt       0.82      0.93      0.88        15

          accuracy                           0.92       315
         macro avg       0.92      0.92      0.92       315
      weighted avg       0.92      0.92      0.92       315
```

Figure 5. Testing Classification Report

The testing classification report, which shows the average performance of each metric and the number of images it was computed on, is shown in Figure 5. This shows that the model excelled in identifying images belonging to simpler classes like ‘Airplane’ and ‘Harbor’, achieving perfect scores, whereas it struggled for visually similar or complex classes like ‘Dense Residential’, ‘Buildings’ and ‘Medium Residential’, highlighting areas that can be improved with further fine-tuning or training with more data to allow for better differentiation. However, the model achieved high scores overall.

3.4 Effectiveness of the Network

The network achieved a 91.8% test accuracy, supported with high F1 scores, precision and recall. This demonstrates strong performance on the UC Merced Land Use Dataset. The training and validation accuracy/loss plots show both smooth and steady convergence, which indicate that the model learned effectively without overfitting.

The network performed well due to a variety of key factors, the first being that the backbone of the network was the InceptionResNetV2 model, which has been trained on an extensive number of images and is reputable for its success in accurately classifying complex images by extracting both small and large patterns in the data. This explains the model’s ability to differentiate subtle patterns between aerial images like ‘beach’ and ‘river.’ Another reason why the model performed well is due to the implementation of data augmentation. Random transformations (e.g., rotations, contrast, zooming) increased the diversity of the training dataset, allowing the model to generalize better and reduce overfitting. Finally, regularization through a dropout rate of 50% was also a key factor in creating an effective model, as it ensured that the model did not rely too heavily on specific neurons during training, to ensure robustness.

The network is also likely to generalize well to new images that are similar (e.g., high-resolution and aerial in format), as the use of the InceptionResNetV2 ensures intricate and reliable feature extraction. However, the model's performance may degrade if the new images differ significantly in terms of lighting or geography (if an area with unseen patterns or different land-use types is given as an input, such as mountains with snow). This is because the model was trained on the UC Merced Dataset and is therefore reliant on it. As a result, adding more diverse data during training, and fine turning the model further would enhance its ability to generalize to make it better suited for a larger variety of input data types, catering to more real-world scenarios.

4 Conclusion

This project involved classifying 2,100 high-resolution RGB aerial images from the UC Merced Land Use Dataset into 21 distinct categories using transfer learning with the InceptionResNetV2 model as the backbone for a customized CNN. Data augmentation, preprocessing and callbacks like early stopping and learning rate reduction were used to optimize training, and a final test accuracy of 92% was achieved indicating that the model performed effectively without overfitting. The training and validation curves demonstrated smooth convergence, while the confusion matrix and classification report highlighted robust performance across most classes, with minor challenges observed in classifying visually similar categories like 'Dense Residentials' and 'Buildings.' This project shows the effectiveness of deep learning in image classification tasks and its reliability with interpreting real-world data.

5 References

[1] "Final Project Overview - ENEL 525 L01 - (Fall 2024) - Machine Learning for Engineers." Accessed: Dec. 18, 2024. [Online]. Available: https://d2l.ucalgary.ca/d2l/le/content/618472/viewContent/6767600/View

[2] "UC Merced Land Use Dataset." Accessed: Dec. 17, 2024. [Online]. Available: https://www.kaggle.com/datasets/abdulhasibuddin/uc-merced-land-use-dataset

[3] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, Feb. 2017, doi: 10.1609/aaai.v31i1.11231.

[4] Code8Soumya, “Unlocking the Power of Vision with Inception-Resnet models: A Journey through Cutting-Edge Deep…,” Medium. Accessed: Dec. 17, 2024. [Online]. Available: https://medium.com/@ssenapati721/unlocking-the-power-of-vision-with-inception-resnet-models-a-journey-through-cutting-edge-deep-4262ef9b28f5

[5] “Improving Inception and Image Classification in TensorFlow.” Accessed: Dec. 17, 2024. [Online]. Available: https://research.google/blog/improving-inception-and-image-classification-in-tensorflow/

[6] D. Yuehua and P. Huilin, “Improved garbage image classification for inception-Resnet-V2,” in *2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML)*, Mar. 2022, pp. 477–481. doi: 10.1109/CACML55074.2022.00087.

[7] “(PDF) Simultaneous Super-Resolution and Classification of Lung Disease Scans.” Accessed: Dec. 18, 2024. [Online]. Available: https://www.researchgate.net/publication/369737866_Simultaneous_Super-Resolution_and_Classification_of_Lung_Disease_Scans

[8] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Jan. 30, 2017, *arXiv*: arXiv:1412.6980. doi: 10.48550/arXiv.1412.6980.

[9] “What Is Cross-Entropy Loss Function?,” 365 Data Science. Accessed: Dec. 17, 2024. [Online]. Available: https://365datascience.com/tutorials/machine-learning-tutorials/cross-entropy-loss/

[10] “What Is Cross-Entropy Loss Function?,” 365 Data Science. Accessed: Dec. 18, 2024. [Online]. Available: https://365datascience.com/tutorials/machine-learning-tutorials/cross-entropy-loss/

[11] “What Is Cross-Entropy Loss Function?,” GeeksforGeeks. Accessed: Dec. 18, 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-cross-entropy-loss-function/

[12] “MachineLearning6.867/Bishop/Bishop - Pattern Recognition and Machine Learning.pdf at master · peteflorence/MachineLearning6.867,” GitHub. Accessed: Dec. 18, 2024. [Online]. Available: https://github.com/peteflorence/MachineLearning6.867/blob/master/Bishop/Bishop%20-%20Pattern%20Recognition%20and%20Machine%20Learning.pdf

[13] D. Karunakaran, “The link between Maximum Likelihood Estimation(MLE)and Cross-Entropy,” Intro to Artificial Intelligence. Accessed: Dec. 18, 2024. [Online]. Available: https://medium.com/intro-to-artificial-intelligence/the-link-between-maximum-likelihood-estimation-mle-and-cross-entropy-599cc1414753

[14] “Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville (z-lib.org).pdf.” Accessed: Dec. 18, 2024. [Online]. Available: http://alvarestech.com/temp/deep/Deep%20Learning%20by%20Ian%20Goodfellow,%20Yoshua%20Bengio,%20Aaron%20Courville%20(z-lib.org).pdf

[15] “Backpropagation in Convolutional Neural Networks,” GeeksforGeeks. Accessed: Dec. 18, 2024. [Online]. Available: https://www.geeksforgeeks.org/backpropagation-in-convolutional-neural-networks/