

MPA-MLF

Final project

MACHINE LEARNING FUNDAMENTALS

04.09.2024

CLASSIFICATION OF ROOM OCCUPANCY

CLARK ALOPH
SAMIA ZEMITI

TEACHERS

BOLCEK JAN, ING.

MARŠÁLEK ROMAN, PROF. ING., PH.D.

Table of content

| | |
|--|--------------|
| 1. INTRODUCTION | - 2 - |
| 2. PROBLEM DESCRIPTION | - 2 - |
| 3. METHODOLOGY | - 3 - |
| 3.1. DATA PREPARATION AND PREPROCESSING | - 3 - |
| 3.1.1. <i>Extraction and Loading of Signal Snapshots</i> | - 3 - |
| 3.1.2. <i>Image Preprocessing and Augmentation</i> | - 3 - |
| 3.2. MODEL ARCHITECTURE..... | - 3 - |
| 3.2.1. <i>Convolutional Neural Network (CNN) Design</i> | - 3 - |
| 3.2.2. <i>Choice of Optimizer and Loss Function</i> | - 3 - |
| 3.3. TRAINING APPROACH | - 4 - |
| 3.3.1. <i>Splitting Data into Training and Validation Sets</i> | - 4 - |
| 3.3.2. <i>Model Training and Hyperparameter Tuning</i> | - 4 - |
| 3.4. EVALUATION METRICS..... | - 4 - |
| 3.4.1. <i>Accuracy, Precision, Recall, and F1 Score</i> | - 4 - |
| 3.4.2. <i>Confusion Matrix Analysis</i> | - 4 - |
| 3.5. PARAMETER CONFIGURATION | - 4 - |
| 4. CHALLENGES ENCOUNTERED | - 5 - |
| 4.1. INITIAL DIFFICULTIES WITH MODEL ACCURACY | - 5 - |
| 4.2. COMPUTATIONAL EFFICIENCY | - 6 - |
| 4.3. PROJECT TIMELINE CONSTRAINTS | - 6 - |
| 5. RESULTS AND DISCUSSION | - 6 - |
| 6. CONCLUSION | - 8 - |
| 7. FUTURE WORK..... | - 8 - |
| 8. CODE GITHUB | - 9 - |
| 9. FIGURES AND TABLES | - 9 - |
| 10. REFERENCES | - 9 - |

1. Introduction

In the realm of Machine Learning and Signal Processing, the capability to accurately classify and interpret signal data to determine room occupancy has profound implications across various domains, including security, energy efficiency, and smart home automation. This report delineates the methodology, experimentation, and findings of our project, tasked with the classification of the number of persons in a room based on 60 GHz signal transmissions. Utilizing a dataset comprised of delay-Doppler domain snapshots, our objective was to develop a model capable of distinguishing among four distinct classes ranging from the presence of machinery alone to the occupancy of up to three persons. These signal snapshots, encapsulating the reflections from targets within the room, serve as the cornerstone of our analysis and model training. Through a meticulous process involving data preparation, model architecture selection, training, and rigorous evaluation, we aimed not only to achieve high accuracy on a withheld test dataset but also to provide insights into the underpinnings of effective signal-based occupancy classification. Figure 1 illustrates an example of the input data used by our model. It depicts a delay-Doppler snapshot of signal reflections, a visual representation of the room's occupancy captured by a signal processing technology.

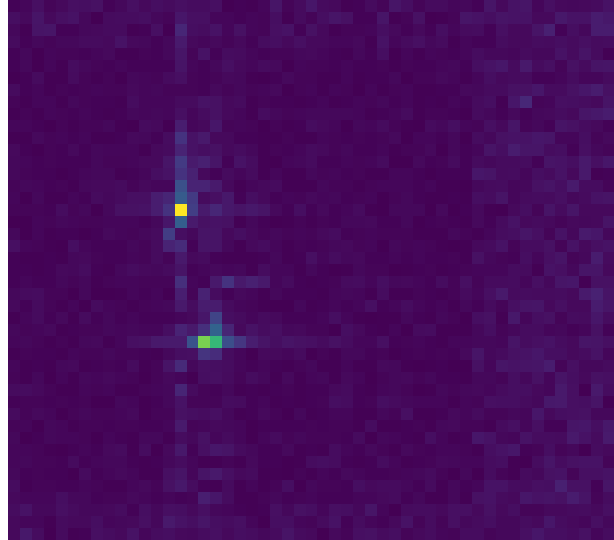


Figure 1 Delay-Doppler Signal Snapshot

2. Problem Description

The problem involves image classification in the delay-Doppler domain. The dataset contains snapshots representing reflections from targets (cf. Figure 1), and the task is **to classify the number of persons present in each snapshot**. This is a challenging problem due to the complexity of signal processing and the potential presence of noise and artifacts in the data.

3. Methodology

3.1.Data Preparation and Preprocessing

3.1.1. Extraction and Loading of Signal Snapshots

The initial step in our project involved extracting and loading signal snapshots from ZIP files stored on Google Drive. These files, `train_data_unlabeled.zip` and `test_data_unlabeled.zip`, contain signal snapshots in the delay-Doppler domain, representing the room's occupancy state. A Python script utilizing the `zipfile` library automated the extraction process, depositing the images into designated training and testing directories for subsequent loading and processing.

3.1.2. Image Preprocessing and Augmentation

To prepare the images for model training, we implemented a function `load_images` to convert images to NumPy arrays, keeping only the RGB channels. This preprocessing step ensures our model receives standardized input. Furthermore, to mitigate overfitting and improve the model's generalization ability, we considered data augmentation techniques such as rotation and flipping. However, due to the nature of our data (signal snapshots in the delay-Doppler domain), preserving the original orientation and scale was deemed crucial for maintaining the integrity of the signal information. Therefore, augmentation strategies were cautiously evaluated but not extensively applied.

3.2.Model Architecture

3.2.1. Convolutional Neural Network (CNN) Design

Our model architecture of choice is a Convolutional Neural Network (CNN), renowned for its efficacy in image recognition tasks. The CNN was designed with multiple convolutional layers followed by max-pooling layers, effectively capturing the spatial hierarchies in the signal snapshots. The convolutional base is succeeded by a flattening layer and a dense layer culminating in a softmax layer for classification. The architecture was meticulously crafted to balance complexity and performance, ensuring adequate capacity to learn from the delay-Doppler domain features without succumbing to overfitting.

3.2.2. Choice of Optimizer and Loss Function

For optimizing our CNN, we selected the Adam optimizer, favored for its adaptive learning rate properties, thereby facilitating faster and more effective model convergence. The learning rate was initially set to 0.001, a value that strikes a balance between rapid learning and the risk of overshooting minima. The loss function employed is categorical cross entropy, appropriate for our multi-class classification task, enabling the model to output a probability distribution over the four possible room occupancy states.

3.3. Training Approach

3.3.1. Splitting Data into Training and Validation Sets

The dataset was randomly split into training and validation sets, with 80% of the data allocated for training and 20% reserved for validation. This split was performed to monitor the model's performance on unseen data during training, thus avoiding overfitting and ensuring the model's generalizability.

3.3.2. Model Training and Hyperparameter Tuning

Model training was conducted over 30 epochs, a duration chosen based on preliminary experiments that balanced between underfitting and overfitting. Training utilized a batch size that allowed for efficient computation while maintaining adequate gradient estimation. Hyperparameters, including the learning rate and the number of filters in convolutional layers, were initially selected based on conventional practices and fine-tuned iteratively based on validation set performance.

3.4. Evaluation Metrics

3.4.1. Accuracy, Precision, Recall, and F1 Score

Model performance was evaluated using accuracy, precision, recall, and F1 score. Accuracy measures the overall correctness of the model across all classes, while precision and recall provide insights into the model's ability to minimize false positives and false negatives, respectively. The F1 score, a harmonic mean of precision and recall, offers a single metric to assess the balance between them, particularly useful in imbalanced datasets.

3.4.2. Confusion Matrix Analysis

The confusion matrix served as a pivotal tool for visualizing the model's performance across different classes, revealing patterns in misclassifications that could inform further model refinement. It was particularly useful for identifying any biases towards specific classes, thereby directing efforts to rebalance the dataset or adjust the model to improve overall accuracy and fairness.

Through this rigorous methodology, we aimed not only to achieve high accuracy on our classification task but also to glean insights into the complexities of signal-based room occupancy detection, paving the way for future advancements in the field.

3.5. Parameter configuration

Table 1. encapsulates the critical parameters that were adjusted to fine-tune our CNN's performance. Each parameter was chosen after several iterations and considering best practices in the field of deep learning. For example, the ReLU activation function was selected for its efficiency and effectiveness in many deep learning models. The Adam optimizer was chosen for its adaptiveness in handling sparse

gradients on noisy problems. The learning rate of 0.001 was determined as a starting point that allows for convergence to a good solution without causing instability in the learning process.

The architecture and hyperparameters of our CNN model were designed to capture the complexities within the delay-Doppler signal snapshots, while also ensuring the model did not overfit to the training data. The chosen configuration reflects a balance between model complexity, computational efficiency, and performance, contributing to the model's high accuracy and generalization ability.

Table 1. Model Architecture and Configuration

| Component | Configuration | Description |
|-----------------------|--|--|
| Convolutional Layer 1 | 32 filters, (3, 3) kernel size, ReLU activation | Captures basic features from input images. |
| Convolutional Layer 2 | 64 filters, (3, 3) kernel size, ReLU activation | Builds on previous features to detect more complex patterns. |
| Convolutional Layer 3 | 128 filters, (3, 3) kernel size, ReLU activation | Extracts high-level features for classification. |
| Max Pooling | (2, 2) pool size | Reduces spatial dimensions after each convolutional layer. |
| Flattening | - | Converts 2D feature maps to 1D feature vectors. |
| Output Layer | 4 units, Softmax activation | Produces a probability distribution across four classes. |
| Optimizer | Adam | Adapts learning rate during training for efficiency. |
| Learning Rate | 0.001 | Set to ensure steady convergence during training. |
| Training Epochs | 30 | Total rounds of training through the dataset. |

4. Challenges encountered

4.1.Initial Difficulties with Model Accuracy

One of the primary challenges we faced early in the project was achieving an acceptable level of accuracy. Initially, our model's performance was suboptimal. A key issue was the complexity of the delay-Doppler signal data, which often contained noise, reflections, and missed targets that could easily mislead the training process. We learned that the signal processing in the delay-Doppler domain required careful handling to ensure that the model could distinguish between subtle patterns indicative

of different occupancy states. Adjusting the convolutional layers and fine-tuning the network to capture the nuances of the signal data better ultimately improved our model's accuracy.

4.2. Computational Efficiency

Another significant obstacle was the computational demands of training our CNN. The training initially took an inordinate amount of time, roughly 30 minutes per execution (and before optimizing our code it was more than an hour...), which impeded our ability to iterate quickly. Switching the runtime environment from a CPU to a T4 GPU, the training time was drastically reduced to less than 3 minutes. That's how we learned that GPUs are specifically designed to handle the parallel processing requirements of deep learning algorithms more efficiently than CPUs. By utilizing the T4 GPU's architecture, we could leverage its multiple cores to perform matrix operations and data processing at a much faster rate, thus accelerating the training process.

4.3. Project Timeline Constraints

Time constraints also posed a significant challenge. Despite our persistent efforts, the optimal methodology and hyperparameter configuration were identified only after the deadline for the Kaggle competition had passed. This was a setback as we could not utilize these improvements to enhance our standing on the Kaggle leaderboard. However, the insights gained post-deadline were still valuable and will be applied in future projects to avoid similar errors.

These challenges were instructive and provided us with a deeper understanding of both the practical


| | | |
|---|----------------|----------------|
|  submission-4.csv Complete (after deadline) · 1h ago | 0.97039 | 0.96872 |
|  submission-2.csv Complete · 21h ago | 0.24687 | 0.25312 |

Figure 2 Kaggle Submission Timelines

and theoretical aspects of machine learning. They have prepared us to better anticipate and mitigate such issues in the future, paving the way for more streamlined and successful project executions.

5. Results and Discussion

Our model's performance on the validation dataset was quantified through several key metrics, each indicative of the model's robustness and accuracy.

In fact, the model achieved an outstanding accuracy of **97.16%**, demonstrating its overall reliability in classifying the number of persons present accurately. The precision of **97.18%** is particularly commendable, as it reflects the model's ability to minimize false positives, an essential aspect for applications where unnecessary alerts are undesirable. Equally impressive is the recall of **97.16%**, which suggests the model's success in correctly identifying the presence of persons. The harmonization

of precision and recall is encapsulated in the F1 score of **97.15%**, asserting the model's balanced performance.

An instrumental part of our evaluation was the analysis of the confusion matrix, which provided an in-depth look at the model's predictive distribution.

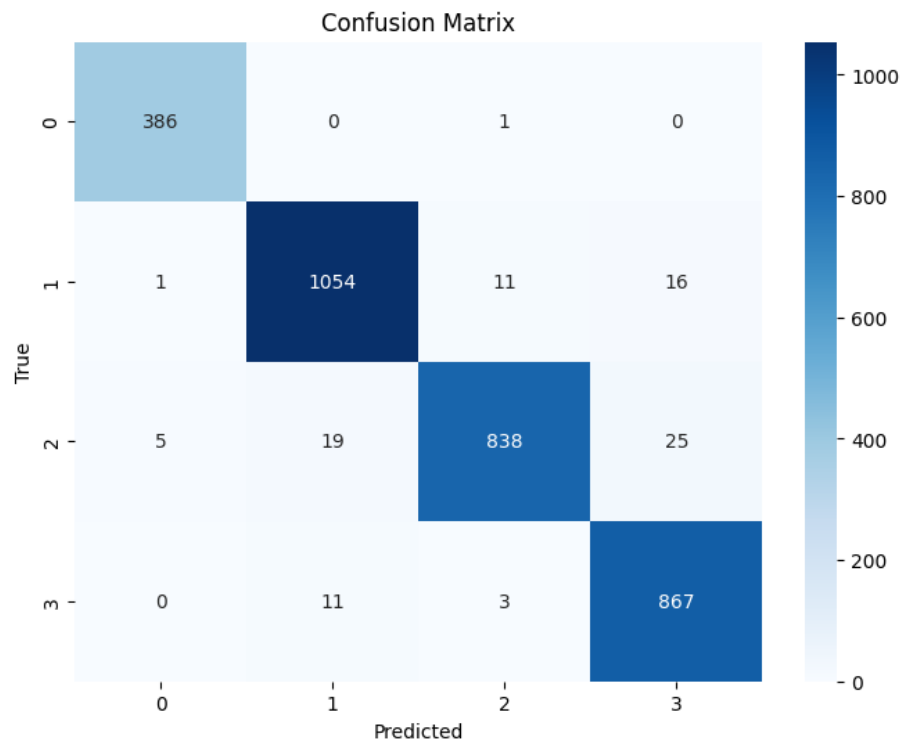


Figure 3 Model Confusion Matrix

The confusion matrix (Figure 2) visualizes the model's performance across different classes. There is a small number of cases where two persons have been mistaken for three and vice versa (as shown by the off-diagonal values of 25 and 3). This could suggest that certain signal patterns reflecting the presence of two or three individuals are somewhat similar, potentially due to the movement or position of the individuals in relation to the signal receiver. These results are a testament to the model's robustness and the effectiveness of the methodologies applied in processing the input data, designing the model architecture, and training the model.

Furthermore, to visually demonstrate the model's discernment abilities, we present an array of signal snapshots alongside their corresponding classification outcomes.

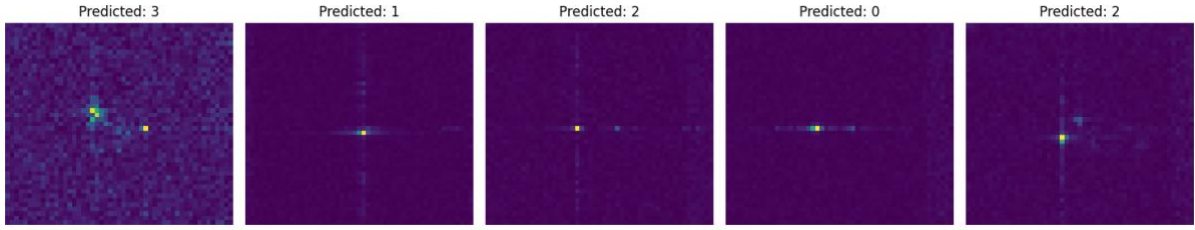


Figure 4 Classification Examples

This collection of snapshots (Figure 3) showcases the model's classification capabilities, presenting several examples of the delay-Doppler images and their respective predicted occupancy states.

In conclusion, the model not only met the project's expectations but also set a precedent for the potential of machine learning in interpreting signal-based data. It stands as a significant stride towards more intelligent, data-driven decision-making in various applications, from smart home management to security and beyond.

6. Conclusion

Our Convolutional Neural Network (CNN) model, through its nuanced interpretation of delay-Doppler domain images, has demonstrated a high level of accuracy in discerning between different occupancy states. The model's performance highlights the success of our approach in addressing the intricate task at hand. From data preparation to strategic model architecture design and vigilant training, we designed a model that meets expectations. Our judicious choice of hyperparameters and computational strategies catalyzed the efficiency of our training process, thereby exemplifying the importance of resource selection in machine-learning workflows.

The experiences have fortified our understanding of the theoretical and practical aspects of machine learning, preparing us for future challenges and endeavors in this dynamic field. Our journey from signals to insights continues, and we are poised at the brink of new discoveries that will undoubtedly propel the field toward more intelligent and autonomous systems.

7. Future Work

In future work, it is essential to investigate techniques for noise reduction and artifact removal to enhance the quality of input data. This can be achieved by exploring denoising techniques, such as autoencoders or variational autoencoders (VAEs), to learn and remove noise from the input signal snapshots. Additionally, signal processing methods, such as filtering or adaptive thresholding, can be investigated to suppress artifacts and enhance the clarity of signal reflections in the data. Furthermore, utilizing domain-specific knowledge and heuristics to develop customized preprocessing techniques tailored to the characteristics of the signal snapshots can significantly improve the overall quality of the input data.

Moreover, collecting additional labeled data is crucial to further train and validate the model on diverse scenarios and environments. This can be accomplished by conducting data collection campaigns in various real-world environments to capture a broader range of scenarios, including different room layouts, lighting conditions, and movement patterns. Collaborating with domain experts to annotate the collected data with ground truth labels ensures an accurate representation of different occupancy levels and configurations. Additionally, augmenting the existing dataset with synthetically generated data or simulated scenarios can supplement the training data and improve the model's generalization capabilities, thereby enhancing its performance in real-world applications.

8. Code GitHub

Our code can be found following this link:

[https://github.com/Izianami/MPA-](https://github.com/Izianami/MPA-MLF/blob/main/Classification_of_room_occupancy_%5BALOPH_ZEMITI%5D.ipynb)

[MLF/blob/main/Classification_of_room_occupancy_%5BALOPH_ZEMITI%5D.ipynb](https://github.com/Izianami/MPA-MLF/blob/main/Classification_of_room_occupancy_%5BALOPH_ZEMITI%5D.ipynb)

9. Figures and Tables

| | |
|--|-------|
| Figure 1 Delay-Doppler Signal Snapshot..... | - 2 - |
| Figure 2 Kaggle Submission Timelines..... | - 6 - |
| Figure 3 Model Confusion Matrix..... | - 7 - |
| Figure 4 Classification Examples | - 8 - |
| Table 1. Model Architecture and Configuration..... | - 5 - |

10. References

- [1] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
- [2] Abadi, M. et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16).
- [3] Chollet, F. (2017). Deep Learning with Python. Manning Publications.
- [4] Sani, N. S., Shamsuddin, I. I. S., Sahran, S., Rahman, A. H. A., & Muzaffar, E. N. (2018). Redefining selection of features and classification algorithms for room occupancy detection. International Journal on Advanced Science, Engineering and Information Technology, 8(4-2), 1486-1493.
- [5] Dai, X., Liu, J., & Zhang, X. (2020). A review of studies applying machine learning models to predict occupancy and window-opening behaviours in smart buildings. Energy and Buildings, 223, 110159.