CSC345: Land Cover Detection using Aerial Imagery

Ross Usher (2010826)

Table of Contents

Introduction 2
Methodology 2
  Support Vector Machine 2
    Feature Extraction 2
    Data Preprocessing 2
    Dimension Reduction 3
    Hyperparameter Selection 3
    Model Training and Evaluation 3
  Convolutional Neural Network 3
    Data Preprocessing 3
    Model Architecture 3
    Model Training and Evaluation 3
Results 4
  Support Vector Machine 4
  Convolutional Neural Network 4
Conclusion 5
Bibliography 6
Appendix 7
    Cumulative Variance Ratio plots for SVM model 7
    Summary of CNN architecture 7
    Accuracy and Loss curves for Training and Validation of the CNN: Initial Model 8
    Accuracy and Loss curves for Training and Validation of the CNN: Using Data Augmentation 8
    Normalized Confusion Matrix for the SVM and CNN models 8

Introduction

In today's digital age, the volume of image data is growing rapidly, thanks to the widespread availability of camera equipment. Teaching computers to identify objects within images holds immense potential in various fields, from healthcare to surveillance. However, automated object detection remains a challenging task for researchers.

This report focuses on land cover detection using aerial imagery, a crucial application in remote sensing. The goal was to develop machine learning algorithms to classify images into different land cover categories. The dataset used in this project originates from the UC Merced Land Use dataset, derived from large-scale urban area imagery (Yang & Newsam, 2010). The dataset comprises 2100 RGB images, available in two resolutions, along with corresponding labels. To classify these images, an analysis approach of implementing two classification methods for the 21 classes and comparing them was chosen. A support vector machine was implemented as well as a convolutional neural network. Accuracies of 60.95% and 81.90% were achieved by the SVM and CNN respectively. While the CNN produce a very good result, the SVM model performed significantly poorer to models that have been trained in other work (Cao et al., 2019; Yaşar & Utku, 2023).

This report outlines the approach, including data preprocessing, feature extraction, model selection, and evaluation. The experimental results are presented, including performance metrics and visualizations, and potential areas for improvement are discussed.

Methodology

For this project, several packages were imported. These include: numpy, matplotlib, tensorflow, sklearn, skimage, seaborn (Abadi et al., 2016; *Array Programming with NumPy | Nature*, n.d.; Hunter, 2007; Pedregosa et al., n.d.; van der Walt et al., 2014; Waskom, 2021). The uses for these packages will be described throughout the report.

Due to computational limitations, the low-resolution images were used for this project. This allowed processing times to be reduced when training the model.

Support Vector Machine

Feature Extraction

Once the image and label datasets had been loaded. A handcrafted feature extractor was used to extract the Histogram of Oriented Gradients (HOG) from the input data.

Data Preprocessing

Once the features had been extracted, the data was split into 3 subsets: training, validation, and testing. This was done using a ratio of 80:10:10 respectively.

Next, the image data for each subset was standardised using the Standard Scaler from sklearn. The labels for each subset were also reduced to 1D using numpy.ravel().

Dimension Reduction

Working with high-dimensional data is not only computationally challenging, but, it also increases the probability of the model overfitting the data. Therefore, principal component analysis was used to reduce the dimensionality of the data. To select the number of components to reduce the data to, a plot of the cumulative variance ratio after PCA was produced. The number of components where the cumulative variance ratio is greater than or equal to 95% was chosen as this will capture a large variance of the data whilst reducing the dimensionality of the data.

Hyperparameter Selection

To select the best hyperparameters for the model, a grid search was done on the validation set. This was done because a grid search can be very computationally consuming so, performing the search on a smaller dataset like the validation set reduces the processing time required. Cross-validation was also used.

Model Training and Evaluation

The model was then trained on the training data using the optimal parameters that were found by the grid search.

The model's performance was measured on both the training and testing data using the accuracy metric. Additionally, a confusion matrix was generated to visualize the model's performance across different land cover classes and identify any misclassifications. The matrix was presented as a heat map (using the seaborn package) to aid visualization.

Convolutional Neural Network

Data Preprocessing

Once the image and label datasets had been loaded, the data was split into three subsets: training, validation, testing. This was done in an 80:10:10 split respectively.

Next, the pixel values for each set were normalised to the range [0,1] by dividing the pixel values by 255.

Model Architecture

The model architecture comprises multiple layers for feature extraction and classification. It starts with three convolutional layers with ReLU activation, each followed by max-pooling to reduce spatial dimensions. Dropout layers are included after each convolutional layer to prevent overfitting. After flattening the feature maps, the model has two fully connected layers with ReLU activation and dropout regularization. The final layer employs softmax activation for multi-class classification with 21 output classes. A summary of the architecture can be found in the appendix of this report.

Model Training and Evaluation

The model was compiled using the Adam optimizer. Since the data is in integer format, the sparse_categorical_crossentropy loss function was used. The accuracy metric was used to measure the proportion of correct classifications that the model predicted.

The model was fitted on the training data for 100 epochs, using the validation data to evaluate the model after each epoch.

The model was then evaluated on the testing data and the accuracy was reported. The accuracy and loss of the model over the training and validation data were plotted which allowed for an analysis of the model's performance and enabled the identification of any issues with the training process such as overfitting.

A confusion matrix was also generated for this model and plotted as a heatmap just like what was done for the previous model. Since both confusion matrices were normalised, a direct comparison could be made between the two models by comparing the matrices.

Results

Support Vector Machine

The first quantitative result that was produced for the SVM model was the plotting of the cumulative variance ratio after PCA dimensionality reduction. Figure 2 shows the plot for the cumulative variance ratio. The number of components required to achieve 95% variance was calculated as 343. The plot can be seen in the appendix of this document.

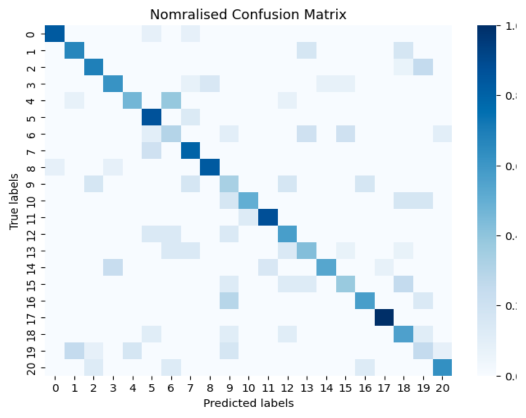
The next quantitative result produced was from the grid search. This found the best parameters for the data to be:

Best Hyperparameters = {'C': 10, 'degree': 2, 'gamma': 'scale', 'kernel': 'rbf'}

Using these hyperparameters to fit the training data, the model achieved a training accuracy of 100% and a testing accuracy of 60.952% (rounded to 3.d.p). This result suggests that there is a large amount of overfitting from the model.

Figure 1

Confusion Matrix for SVM Model.



The normalised confusion matrix produced for the model can be seen in Figure 1. The heatmap shows a diagonal line from the top left of the matrix to the bottom right. This means that the data predicted a lot of the classes correctly; however, there is a lot lighter shaded cells throughout the matrix which shows that the model made quite a few inccorect classifications. This is consistent with the 60.952% accuracy that was achieved on the testing data.

Convolutional Neural Network

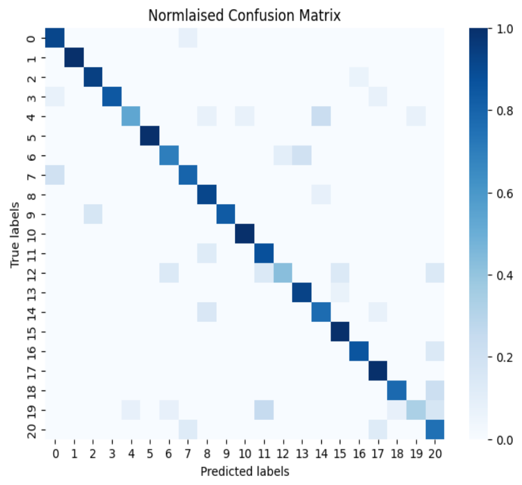
The first quantitative result produced by the CNN model is the accuracy of the model on the training and testing dataset. When evaluated, the model achieved an accuracy of 99.17% on the training dataset and 71.90% on the testing dataset. This shows that there is some overfitting going on. This was further confirmed by analysing the accuracy and loss curves produced for the training and validating datasets. The curves show that the validating accuracy plateaus whilst the training accuracy continued to climb. Also, the validating loss plateaus whilst the training loss continues to fall. The plots of the curves can be seen in the appendix of this report. The plots of the curves can be found in the appendix of this report.

To try to combat the over fitting that was present in the model. Data augmentation was applied to the training data. This added variance to the training data which helped the model to adapt to unseen data, thus, reducing the amount of overfitting present in the model. The data augmentation consisted of applying at random a combination of rotating, shifting, and flipping the image.

After applying data augmentation, the model was fitted onto the training data again. This time, the model was trained for 150 epochs. The model achieved an accuracy of 92.38% on the training data and an accuracy of 81.90% on the testing data. This is a fair amount of improvement on the accuracy of the model. Also, the training and validation curves show a major improvement in terms overfitting. Both the training and validation curves climb together while both loss curves fall together. This shows a great improvement in the fitting of the model and pushing the accuracy up to 81.90% is a great result. The plots of the curves can be found in the appendix of this report.

Figure 2

Confusion Matrix for CNN Model.



The normalized confusion matrix for the model was then produced. This can be seen in Figure 2. As expected, there are many more darker cells in the diagonal line than in the SVM model's confusion matrix. There also seems to be fewer coloured cells outside of the diagonal line than in the SVM model's confusion matrix. This is also expected as the CNN model has a higher accuracy than the SVM model. This confusion matrix is consistent with all the feartues you would expect from a model performing at 81.90% accuracy.

Conclusion

The methodology involved utilizing two machine learning algorithms, Support Vector Machine (SVM) and Convolutional Neural Network (CNN), to classify land cover categories using aerial imagery. SVM exhibited perfect training accuracy but suffered from overfitting, while CNN showed superior performance despite slight overfitting. Data augmentation notably mitigated CNN's overfitting, leading to improved generalization. However, both methods have shortcomings: SVM's tendency to overfit necessitates careful regularization, while CNN's architecture complexity may require further optimization to balance accuracy and generalization. Future improvements could involve exploring advanced regularization techniques for SVM and fine-tuning CNN architectures to enhance both accuracy and robustness in land cover detection tasks.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016, May 27). *TensorFlow: A system for large-scale machine learning*. arXiv.Org. https://arxiv.org/abs/1605.08695v2

Array programming with NumPy | Nature. (n.d.). Retrieved 6 May 2024, from https://www.nature.com/articles/s41586-020-2649-2

Cao, C., Dragićević, S., & Li, S. (2019). Land-Use Change Detection with Convolutional Neural Network Methods. *Environments*, *6*(2), Article 2. https://doi.org/10.3390/environments6020025

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., & Cournapeau, D. (n.d.). Scikit-learn: Machine Learning in Python. *MACHINE LEARNING IN PYTHON*.

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., & scikit-image contributors. (2014). scikit-image: Image processing in Python. *PeerJ*, *2*, e453. https://doi.org/10.7717/peerj.453

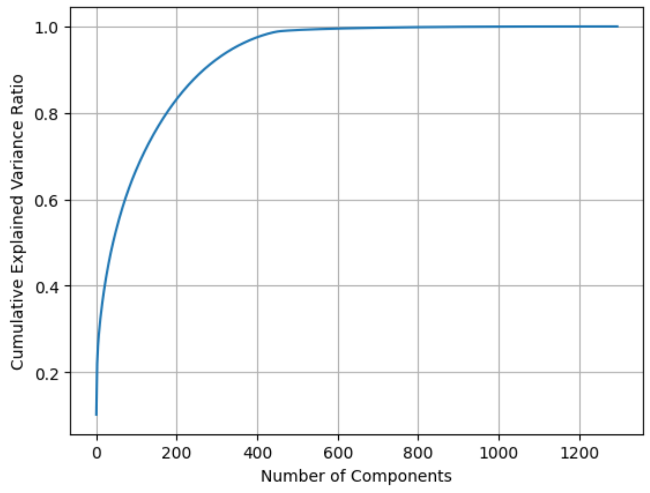
Waskom, M. L. (2021). seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Yang, Y., & Newsam, S. (2010). Bag-of-visual-words and spatial extensions for land-use classification. *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 270–279. https://doi.org/10.1145/1869790.1869829

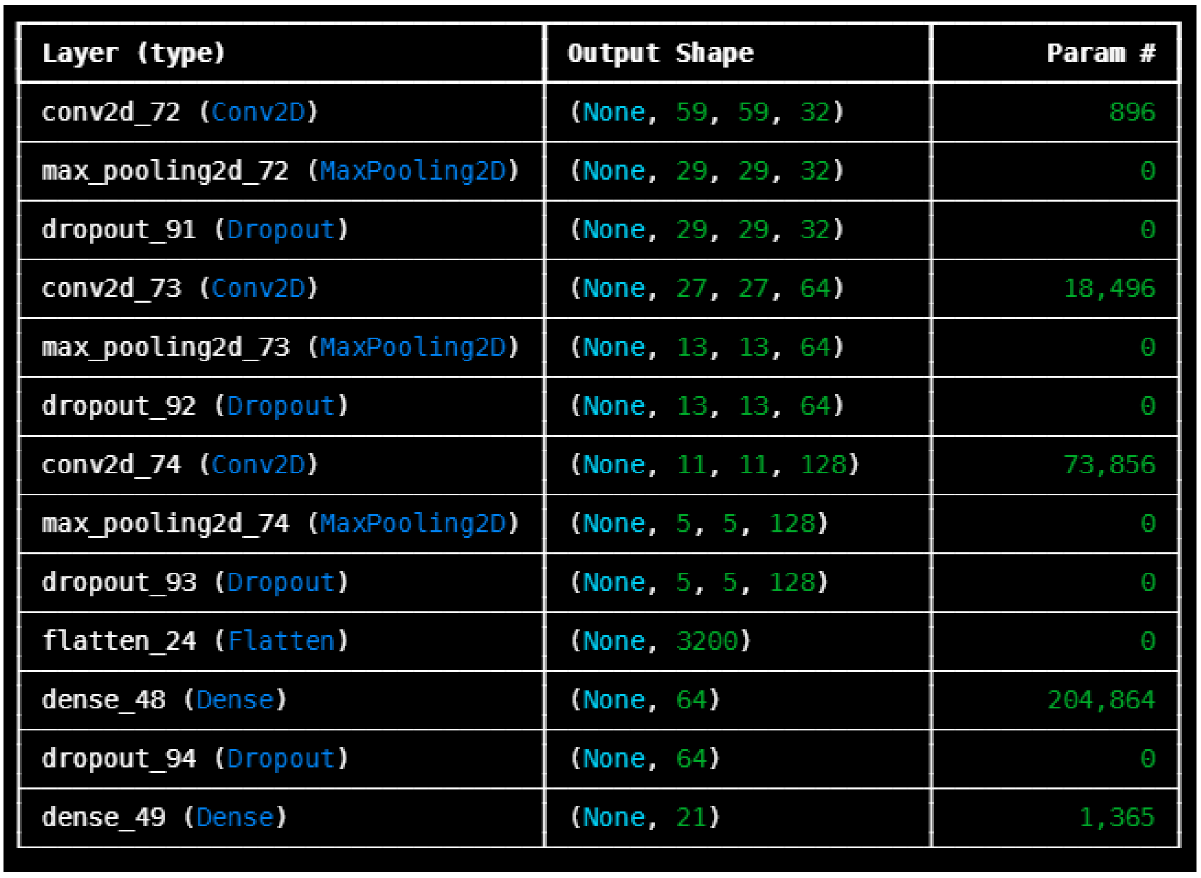
Yaşar, F., & Utku, S. (2023). Performance Comparison of CNN Based Hybrid Systems Using UC Merced Land-Use Dataset. *Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi*, *25*(75). https://doi.org/10.21205/deufmd.2023257516

Appendix

Cumulative Variance Ratio plots for SVM model

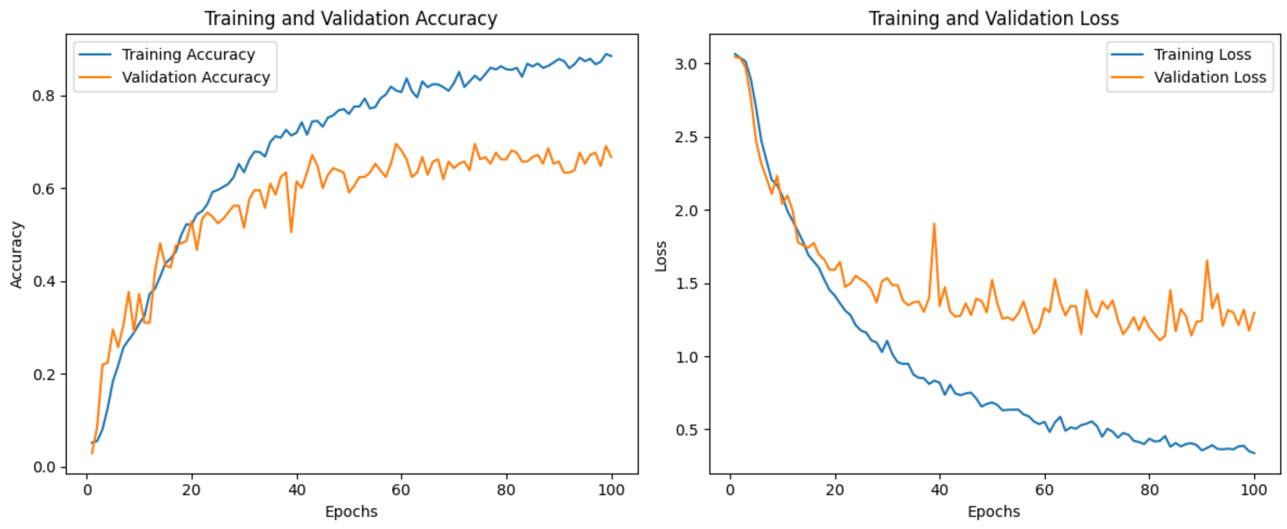


Summary of CNN architecture

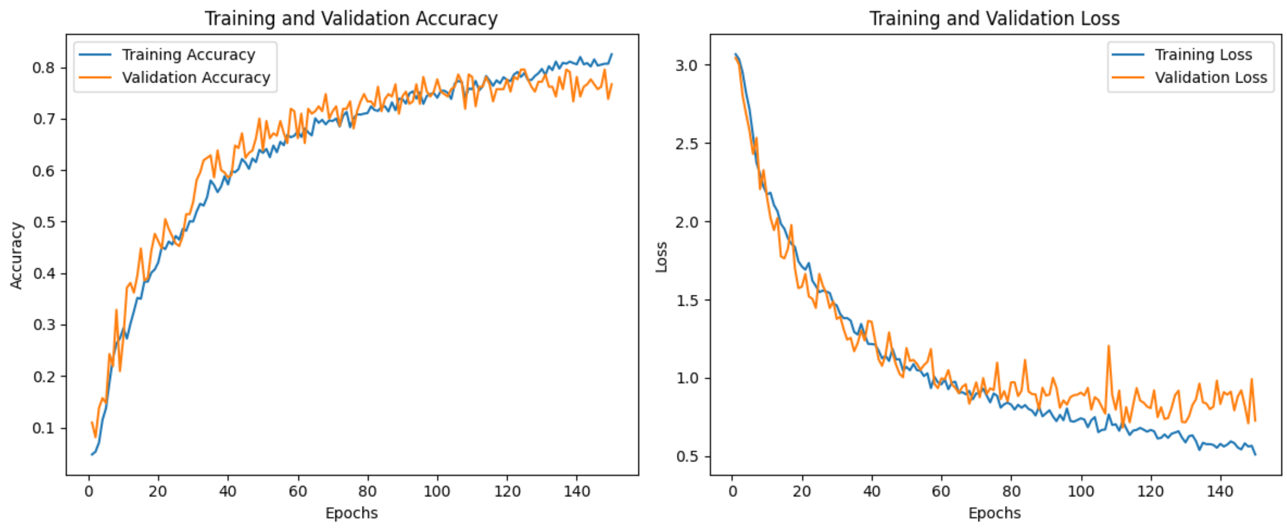


Layer (type)	Output Shape	Param #
conv2d_72 (Conv2D)	(None, 59, 59, 32)	896
max_pooling2d_72 (MaxPooling2D)	(None, 29, 29, 32)	0
dropout_91 (Dropout)	(None, 29, 29, 32)	0
conv2d_73 (Conv2D)	(None, 27, 27, 64)	18,496
max_pooling2d_73 (MaxPooling2D)	(None, 13, 13, 64)	0
dropout_92 (Dropout)	(None, 13, 13, 64)	0
conv2d_74 (Conv2D)	(None, 11, 11, 128)	73,856
max_pooling2d_74 (MaxPooling2D)	(None, 5, 5, 128)	0
dropout_93 (Dropout)	(None, 5, 5, 128)	0
flatten_24 (Flatten)	(None, 3200)	0
dense_48 (Dense)	(None, 64)	204,864
dropout_94 (Dropout)	(None, 64)	0
dense_49 (Dense)	(None, 21)	1,365

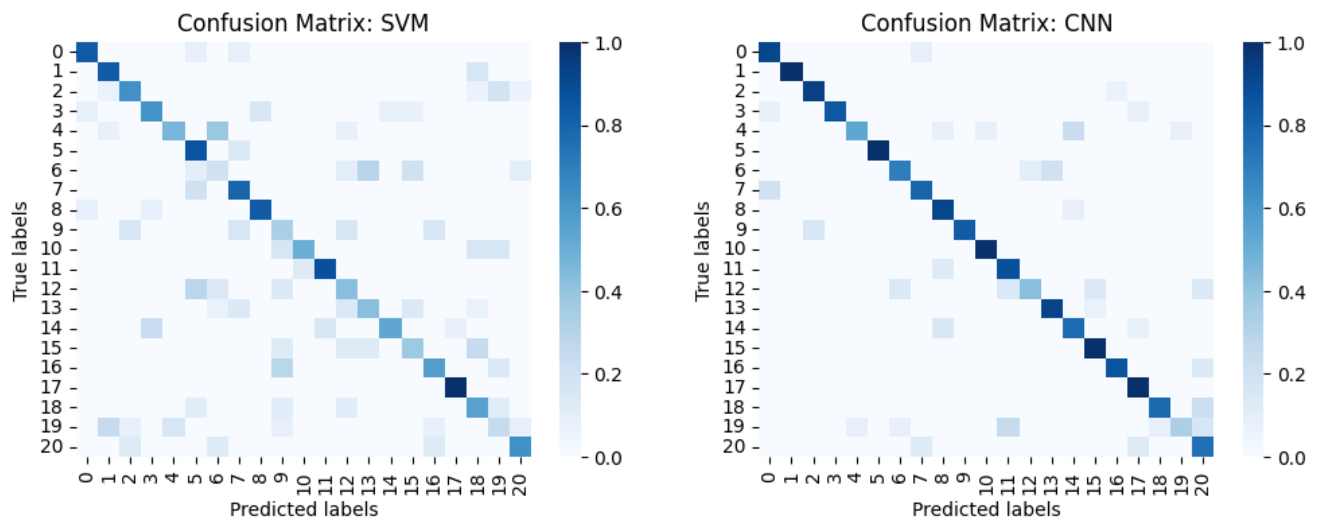
Accuracy and Loss curves for Training and Validation of the CNN: Initial Model



Accuracy and Loss curves for Training and Validation of the CNN: Using Data Augmentation



Normalized Confusion Matrix for the SVM and CNN models


Confusion Matrix: SVM
True labels
Predicted labels
Confusion Matrix: CNN
True labels
Predicted labels