Assignment 2 – Case Study for Implementation

Group: 3

Team Member	Student ID	Contributions
Anamika Valappil	23103632	Came up with the idea of how to implement it
Vaishali Lalit	23104512	Implemented key functionalities for the training pipeline ensuring the effectiveness and efficiency of the system.
Lipsa Sahu	23103186	Led the debugging process of the entire code pipeline and analyzed the dataset for better performance.
Shrutik Kharkar	23103054	Implemented the training pipeline for the model.
Sanjeeta	23101123	Documented the report and analyzed the model results on different hyperparameters.
Shrutik Kharkar	23103054	Implemented the graphical analysis of the model to understand the performance metrics of each of the experiments conducted.
Lipsa Sahu	23103186	Documented the report with all the recorded results and the graphs obtained.
Shrutik Kharkar	23103054	Suggested the idea of having a Siamese network, parallel architecture feeding two inputs and then taking a weighted average of both the models for inference. One of the architectures is fed with the original image and the other architecture is fed with preprocessed feature-enhanced images.
Vaishali Lalit	23104512	Proposed starting with a new model and incorporating layers inspired by InceptionV3, particularly the inception and reduction blocks. This strategy aims to widen the network while maintaining a shallower depth to achieve optimal performance. Additionally, leveraging residual networks, renowned for their superior performance and architectural novelty, further enhances the model.
Sanjeeta	23101123	Suggested to utilize any pre-trained model, remove the top layers, and add additional layers such as batch normalization. Design a custom loss function tailored to the dataset to reduce false positive outcomes and leverage the advantages of pre-trained models.
Anamika Valappil	23103632	Suggested model influenced by autoencoder models with encoder-decoder layers, for learning complex features from the images-implemented model.
Lipsa Sahu	23103186	Incorporating any less compressed CNN and residual network, this facilitates the training of very deep networks by allowing gradients to flow directly through the network without significant degradation.
Anamika Valappil	23103632	Developed the web application, enabling users to upload pictures and receive instant categorization based on the uploaded trained algorithm model.

AlexNet-inspired Autoencoder: Image Classification

Anamika Valappil
School of Computer Science
University of Galway
Galway, Ireland
A.valappil1@universityofgalway.ie

Lipsa Sahu
School of Computer Science
University of Galway
Galway, Ireland
L.sahu1@universityofgalway.ie

Sanjeeta
School of Computer Science
University of Galway
Galway, Ireland
S.sanjeeta1@universityofgalway.ie

Shrutik Kharkar
School of Computer Science
University of Galway
Galway, Ireland
S.kharkar1@universityofgalway.ie

Vaishali Lalit
School of Computer Science
University of Galway
Galway, Ireland
V.lalit1@universityofgalway.ie

Abstract— **This report introduces a custom-built image classification model designed to identify various road surfaces, a key innovation for autonomous vehicle technology and infrastructure management. The model is a unique blend of AlexNet's powerful feature extraction and an autoencoder structure. It starts with an encoder, drawing on AlexNet's approach to distill essential details from images, and compresses them into a simplified representation. A decoder then reconstructs the image from this compressed form, which enables the model to learn the precise feature detection needed for accurate classification.**

Our tailored autoencoder is adept at handling the provided labeled dataset for both reconstructing images and identifying road surfaces. By including specific modifications to the standard AlexNet architecture, the model has been customized to serve as the company's proprietary technology. The performance of the model is carefully evaluated using appropriate metrics, and potential challenges in real-world scenarios are addressed, ensuring the company is well-informed of the model's strengths and limitations.

I. Introduction

With the evolution of autonomous vehicles and the growing need for smart infrastructure, the ability to accurately recognize and differentiate road surfaces has become increasingly vital. This capability is not only central to the safety and efficiency of autonomous navigation systems but also to the effective management of road maintenance and traffic control. As such, the need for advanced computational models that can reliably perform these tasks is important [4]. To meet this demand our team developed a machine-learning model capable of identifying various road surfaces from image data.

This report outlines the process of creating a novel model that extends the well-established principles of the AlexNet architecture, integrating them into an encoder-decoder framework commonly found in autoencoders. Our approach capitalizes on the strengths of AlexNet's convolutional layers for feature extraction while leveraging the autoencoder's capacity for image reconstruction to facilitate robust classification. The data provided by the client, already partitioned into training, validation, and testing sets, enabled a structured approach to the model's development and subsequent evaluation. Through iterative training and meticulous testing, we have refined the model to meet the desired performance criteria.

In the subsequent sections, we delve into the literature that informed our model's design, articulate the rationale behind our architectural choices, and present a comprehensive evaluation of the model's performance. Additionally, we will discuss the model's potential limitations in real-world applications, providing the client with a transparent and thorough understanding of the technology.

II. Literature Review

A. *A Novel Progressive Image Classification Method Based on Hierarchical Convolutional Neural Networks*

Image classification is the basis of computer vision where deep learning techniques are the soul of high accuracy and effectiveness. Over time, several deep learning models have been suggested to address the hardship of image classification from these models, the Convolutional Neural Network (CNN) is one, while AlexNet, VGG16, ResNet, and DenseNet are some other ones.

According to the paper published by Cheng Li, Fei Miao, and Gang Gao (2021) [1] a hierarchical convolutional neural network (HCNN) was proposed, which was capable of image classification in a progressive manner. The HCNN model is a deep learning model that is capable of extracting visual features from images with the help of multiple sub-networks with different depths allowing for the hierarchical and progressive feature extraction process. This approach allows the model to classify images concerning their level of difficulty by adopting sub-networks that are simpler for the easy-to-classify images and more complex for the challenging ones.

One of the major strengths of the HCNN method is the multi-class joint loss function that improves the classifier power of the model by lowering the distance between features of samples from the same category and increasing the distance between features of samples from different categories. The next step is to incorporate this loss function into the hierarchical structure of HCNN, and overall it leads to better performance than the performance of advanced models and ensemble learning methods.

The scalability of HCNN is also of great importance since the model makes the dynamic changes in the type and magnitude of sub-networks which then results in more effective details in the aspect of sub-networks technology. This flexibility enables HCNN to dynamically adjust its weight to various requirements of image classification tasks. The advantage lies in that it creates more possibilities for deep learning-driven image classification to be applied to many tasks.

To wrap it up, the research by Cheng Li, Fei Miao, and Gang Gao (2021) [1] in HCNN shows us an image classification milestone through its hierarchical structure, step-by-step feature extraction method, and better performance than the other models that existed.

B. *SMD-Net: Siamese Multi-Scale Difference-Enhancement Network for Change Detection in Remote Sensing*

Change detection in remote sensing as an important task has various ranges of applications from urban planning and environmental monitoring to disaster management. The technology itself has changed from the old-fashioned pixel and object detection methods to more sophisticated deep learning techniques because of the constant progress in satellite image resolution and developments in computing power.

Traditional Methods: Previously used change detection methods were wholly reliant on pixel-based and object-based computing. Pixel-based techniques, which used image differencing, PCA, and CVA as the basis of their processing, were simple but could not usually handle the spatial pattern complexity and noise in the data. Compared to pixel-based techniques, object-level methods took a step forward by parsing the images into semantic objects and then assessing the changes at this higher level. This resulted in a better context and noise reduction but at the cost of increased complexity and the need for domain expert inputs.

Deep Learning Revolution: The emergence of deep learning, which included Convolutional Neural Networks (CNNs), was a breakthrough change that led to the development of remote sensing. Such models, capable of hierarchical feature encoding, have displayed superiority over the previous techniques in varied tasks that involve object detection, semantic segmentation, and image classification. On the other hand, implementing these models for change detection has its difficulties, such as the interwoven nature of bi-temporal features and the large amount of complex scene changes over extended periods.

Siamese Networks in Change Detection: The Siamese network architecture being networked in parallel form by branches and merging the features at some point is being widely used by researchers as a way to overcome those difficulties [2]. They are used in accurate comparisons of features from bi-temporal images thus making them the best application area in change detection tasks. Models like FC-EF, FC-Siam-Conc, and FC-Siam-Diff have shown promising results by exploiting Siamese structures for more effective feature comparison and fusion.

Performance Evaluation: SMD-Net is notably superior on multiple benchmark datasets, providing a significant enhancement in accuracy and detail preservation in the change detection results, getting higher than the latest methods like FC-Siam-Diff and SNUNet-CD. The algorithm is indeed superior in processing small objects, preserving the whole object, and precisely depicting boundaries among objects that compete in the change detection methodology arena.

Conclusion: SMD-Net undoubtedly is a breakthrough recently that came to be a part of applying the deep learning method to change detection in remote sensing. Taking into account its special multi-scale difference enhancement, which incorporates specialized modules and levels of processing at the top and bottom, this solution targets the core difficulty of change detection accuracy and clarity in the domain of high-resolution remote sensing images.

C. *Seismic fault detection using an encoder–decoder convolutional neural network with a small training set*

Fault identification is an essential responsibility in geoscience and earthquake engineering in terms of prediction of the probability and assessment of the threat of seismic activities. Traditional approaches to fault detection generally depend on the visual examination of seismic data, which consumes lots of time and suffers from subjectivity. In the last few years, a great concern has been demonstrated by making use of deep learning approaches, for example, convolutional neural networks (CNNs) for automatic anomaly detection in seismic data sets.

"Seismic fault detection using an encoder-decoder convolutional neural network with a reduced training data set" - the study by Shengrong, Li & Changchun, Yang & Hui, Sun & Zhang, Hao. (2019) [3] introduces a new method that uses the structure of an encoder-decoder CNN to detect seismic faults even with a small training dataset. Such an approach is extra beneficial in scenarios where there is a problem of either procuring a massive labeled and ready-to-use dataset or the price of gathering this data.

The application of deep learning for the detection of seismic faults has yielded encouraging outcomes. For instance, trained a convolutional neural network (CNN) to solve the problem of the detection of faults in seismic images during experiments by using hierarchical features. Same as this, proposed a deep learning framework for anomaly detection by utilizing the combination of CNNs and RNNs to capture the spatial and temporal irregularities in seismic data.

The architecture used in the present study, that is encoder-decoder, has been proven to be effective in many image-related tasks – for example, segmenting and reconstructing images. By using this architecture favorably in fault detection, the model can also learn to encode appropriate features from the input data and decode them to easily identify fault patterns accurately.

Though the introduction of deep learning for seismic fault detection is successful, still the challenges are not resolved, especially where the data is limited. Overfitting, issues with generalization, and model interpretability are fundamental challenges that researchers commonly face while working on neural networks for fault detection tasks. With that said, it is imperative to find the solutions for these issues to make automated fault detection systems as robust and as reliable in real-life applications as possible.

III. Proposed model

Our model innovatively blends the core principles of AlexNet with the versatile functionality of an encoder-decoder architecture, resulting in a refined autoencoder tailored for image classification purposes. At its core, the model embarks on its computational journey with an encoder network. This network draws inspiration from the convolutional prowess of AlexNet, renowned for its effectiveness in feature extraction. By navigating through the encoder's layers, the model adeptly identifies and isolates important features from the input images, transforming these complex visual inputs into a distilled form known as the latent space representation.

A. Encoder: Inspired by AlexNet

AlexNet Foundation: AlexNet, a convolutional neural network (CNN) architecture, is renowned for its ability to identify and extract complex features from images. It consists of several convolutional layers, followed by max pooling and normalization layers, culminating in fully connected layers. Our model's encoder draws inspiration from these layers, particularly focusing on the convolutional stages for their feature extraction capabilities.

Feature Extraction and Compression: In the encoder phase, the model processes the input images through a series of convolutional layers that mimic AlexNet's structure. These layers filter the images to capture essential visual features, such as edges, textures, and patterns, that are crucial for classification. As the data passes through these layers, it undergoes spatial reduction and feature amplification, effectively compressing the image information into a dense, feature-rich representation[5].

B. Latent Space Representation

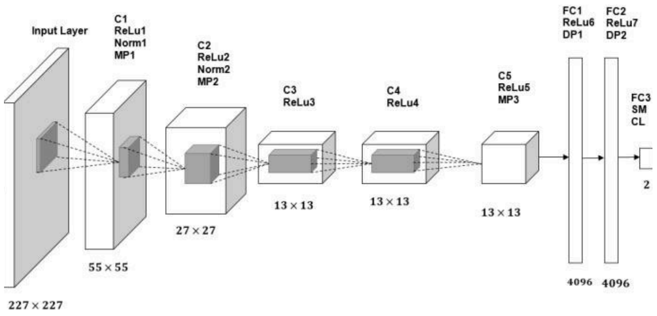
Compressed Feature Space: The encoder's output is a compact, latent space representation of the original input. This latent space is a lower-dimensional embodiment of the image's features, retaining only the most salient information necessary for reconstruction and classification. It serves as the bridge between the encoding and decoding phases, encapsulating the critical data in a more manageable form.

C. Decoder: Mirroring the Encoder

Reconstruction Process: The decoder part of the autoencoder mirrors the encoder's architecture in reverse. It takes the compressed latent representation and gradually reconstructs the original image. This is achieved through a series of up-sampling and convolutional layers that expand the compressed data back into its original spatial dimensions[4].

Learning through Reconstruction: The process of reconstructing the original image from the compressed representation forces the model to learn efficient encoding of the input data. By minimizing the reconstruction error—the difference between the original image and its reconstruction—the model fine-tunes its parameters to preserve essential information during the encoding phase.

The novelty of this model lies in its unique integration of AlexNet's proven feature extraction capabilities with an encoder-decoder architecture, typically found in autoencoders, tailored for image classification. This hybrid approach allows the model to efficiently distill and compress key features from images into a compact latent space representation through the encoder, derived from AlexNet's architecture. The decoder then reconstructs the image from this representation, enabling the model to learn detailed feature distinctions necessary for accurate classification. This dual process of reconstruction and classification introduces a novel methodology for image-based tasks, enhancing both the interpretability and accuracy of the model in distinguishing various road surfaces. The adaptation adds a layer of innovation by leveraging the strengths of both architectures to address complex classification challenges uniquely.



Model Architecture:

The architecture of the model combines a convolutional neural network with an autoencoder framework for the task of image classification. Technical breakdown of the architecture is as follows:

Encoder: The encoder is designed to capture and condense the informational essence of the input images. It begins with an input layer designed to process images of size 224x224 pixels with 3 channels (RGB).

First Convolutional Layer: This layer utilizes 96 filters with a large kernel size of 11x11 to capture the low-level features such as edges and simple textures. A stride of 4 is used to reduce the spatial dimensions right from the start, and ReLU (Rectified Linear Unit) activation introduces non-linearity, enabling the model to learn complex patterns.

First Max Pooling Layer: A 3x3 pool size with a 2-stride is used to perform down-sampling, reducing the feature map's dimensions by half, effectively concentrating the features and reducing computational load.

Second Convolutional Layer: Increases the depth with 256 filters of a 5x5 kernel size, with the 'same' padding ensuring the spatial dimensions are preserved. This layer extracts more complex features while maintaining the spatial hierarchy.

Second Max Pooling Layer: Another 3x3 pool size with a 2-stride further reduces the feature map size, maintaining the model's focus on the most prominent features.

Additional Convolutional Layers: Three more sets of convolutional layers follow, each with a 3x3 kernel size but increasing filter depth (384, 384, and 256). These layers are pivotal for capturing high-level features within the image, such as complex textures and shapes.

Final Encoder Convolutional Layer: Uses 256 filters with a 3x3 kernel, with 'same' padding and a 1-stride, providing a refined set of feature maps representing the encoded state.

Decoder: The decoder's role is to reconstruct the original image from the encoded representation and output the classification probabilities.

Conv2D Transpose Layers: These layers act as the reverse of convolutional layers. The model uses transposed convolutional operations with 256, 384, 384, and 256 filters respectively to perform the up-sampling. This restores the spatial dimensions that were reduced during encoding.

Upsampling Layers: Utilized after certain transposed convolutions to further increase the size of the feature maps, bringing them closer to the original image size.

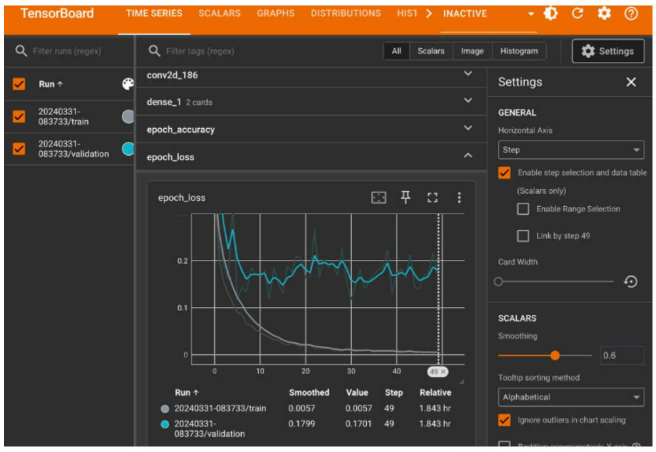
Final Conv2D Transpose Layer: This layer applies a final up-sampling with a 96-filter transposed convolution to nearly restore the original image size.

Flatten Layer: The multi-dimensional tensor is flattened to a single vector, which is necessary for the fully connected layers that follow.

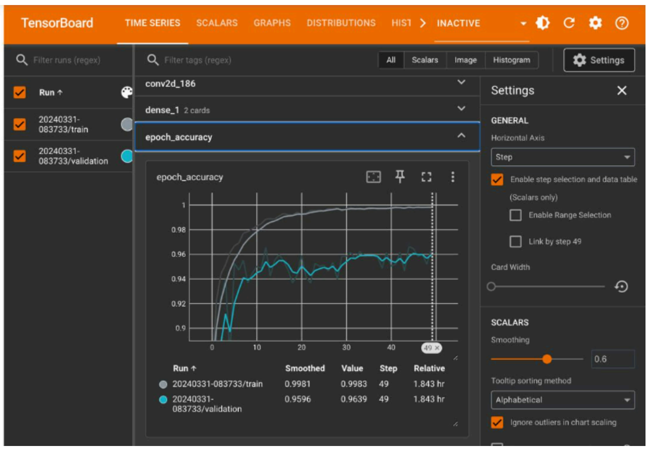
Dense Layers: A series of dense (fully connected) layers follows. The first two have 4096 units each and use ReLU activation. These dense layers are crucial for integrating global information from the feature maps.

Output Layer: The final dense layer with 3 units corresponds to the number of classes, using a softmax activation function to output a probability distribution over the classes.

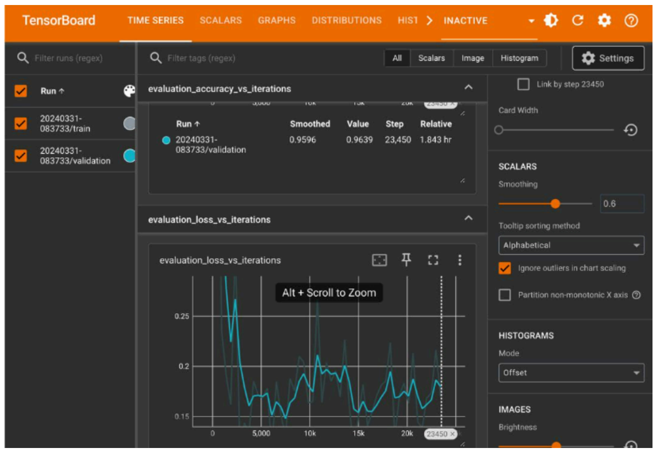
Tensorboard Graphs showing the training and validation loss and accuracy over epochs:



Run ↑	Smoothed	Value	Step	Relative
20240331-083733/train	0.0057	0.0057	49	1.843 hr
20240331-083733/validation	0.1799	0.1701	49	1.843 hr



Run ↑	Smoothed	Value	Step	Relative
20240331-083733/train	0.9981	0.9983	49	1.843 hr
20240331-083733/validation	0.9596	0.9639	49	1.843 hr



Run ↑	Smoothed	Value	Step	Relative
20240331-083733/validation	0.9596	0.9639	23,450	1.843 hr

Inference: Autoencoder model trained on the entire dataset and then the encoder is used for the model inference for the classification task, this encoder-decoder architecture facilitates the model to capture the complex features from the images, and enable encoder alone to perform better on the classification task.

IV. Data analysis

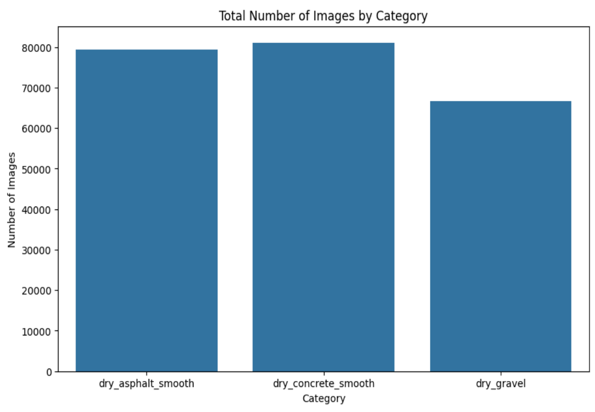


Fig 1: Dataset overview by category

The dataset given by the client is split into three categories, characterized by surface types: dry_asphalt_smooth, dry_concrete_smooth, and dry_gravel. These represent different textures and have been compiled to load into models that are capable of distinguishing between these surface types accurately.

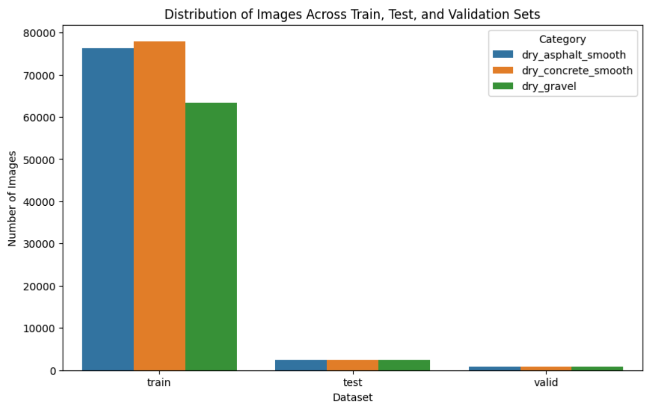


Fig 2: Distribution of images and imbalance across different categories

After investigating the dataset, it showed that there is an unbalanced class ratio, where dry_gravel class is a minority and dry_concrete_smooth and dry_asphalt_smooth classes account for the most.

The total number of images in the train set is distributed as follows: 76,231 for dry_asphalt_smooth, 77,938 for dry_concrete_smooth, and 63,417 for dry_gravel. In test: dry_asphalt_smooth has 2351 images, dry_concrete_smooth has 2351 images and dry_gravel has 2351 images. Whereas, in the validation set: dry_asphalt_smooth has 821 images, dry_concrete_smooth has 821 images and dry_gravel has 821 images.

This distribution suggests an overall imbalance ratio of 0.82 when considering the minimum to maximum class representation, which is particularly evident within the training set with a similar imbalance ratio of 0.81. In contrast, the test and validation sets are perfectly balanced, each having an equal number of images across categories, denoted by an imbalance ratio of 1.00, thus ensuring unbiased evaluation metrics.

If we don't balance the training set, the model might end up biased, favoring the more common classes. Without proper pre-processing, the model could struggle to recognize less frequent classes like dry gravel, both in real-life situations and in the balanced testing and validation sets.

Before we started training the model, we made sure to prepare our dataset with several preprocessing steps to get the images just right. Contrast plays a big role in telling different surface textures apart, so we amped up the contrast in the original images by 50%. This move was crucial for bringing out the unique textural details of each surface, helping the model to pick up on those key features it needs to spot the differences accurately.

Another crucial step in the preprocessing pipeline involved resizing the images to a uniform resolution of 224x224 pixels using the Lanczos resampling algorithm. Lanczos, known for its ability to preserve image quality during the resize process, is particularly adept at maintaining the clarity of edges, which is essential for the model to detect subtle differences in surface textures between categories.

Once preprocessed, the images were fed into the neural network model in batches. The ImageDataGenerator played a dual role here by not only augmenting the data but also streaming it directly to the model during training. This approach is memory-efficient as it eliminates the need for loading the entire dataset into memory at once, which can be particularly advantageous when working with large image datasets.

The combination of contrast enhancement, and advanced resizing techniques, provided a comprehensive approach to data preprocessing and ensured that the dataset was not only robust and diverse but also primed for optimal ingestion by the convolutional neural network model for training.

V. Experiments and Results

The experiments aimed to evaluate the performance of convolutional neural network models on a balanced dataset of 7,050 images across three categories: dry_asphalt_smooth, dry_concrete_smooth, and dry_gravel.

Specifically, the models tested include the AlexNet model with the original images, an AlexNet model with a superimposed image, and the proposed model trained with various optimization algorithms and learning rates.

AlexNet Model with original images:

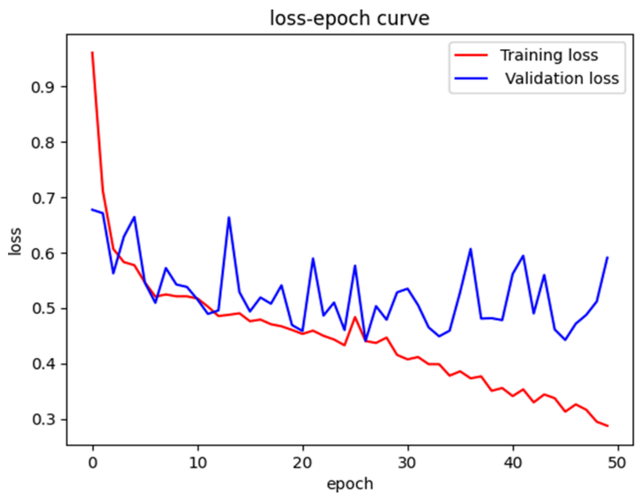


Fig 3: Loss-Epoch Curve for AlexNet Model with Original images

The loss curve for the AlexNet model with original images demonstrates a good fit between the training and validation sets. The training loss shows a consistent decline, closely followed by the validation loss, indicating that the model is generalizing well without significant overfitting.

The AlexNet model with original images yielded an overall accuracy of 80%.

Category	**Precision**	**Recall**	**F1-Score**	**Support**	**dry_asphalt_smooth**	**dry_concrete_smooth**	**dry_gravel**
dry_asphalt_smooth	0.72	0.78	0.75	2350			
dry_concrete_smooth	0.88	0.75	0.81	2350			
dry_gravel	0.83	0.87	0.85	2350			
dry_asphalt_smooth					1833	210	307
dry_concrete_smooth					454	1768	128
dry_gravel					267	31	2052

Table 1: Classification report and confusion matrix for AlexNet Model with Original images

AlexNet Model with Superimposed Images:

The AlexNet model with superimposed images yielded an overall accuracy of 81%.

Category	**Precision**	**Recall**	**F1-Score**	**Support**	**dry_asphalt_smooth**	**dry_concrete_smooth**	**dry_gravel**
dry_asphalt_smooth	0.72	0.79	0.75	2350			
dry_concrete_smooth	0.86	0.82	0.84	2350			
dry_gravel	0.86	0.83	0.84	2350			
dry_asphalt_smooth					1846	255	249
dry_concrete_smooth					355	1919	76
dry_gravel					349	46	1955

Table 2: Classification report and confusion matrix for AlexNet Model with Superimposed images

The original AlexNet model provided strong performance, showcasing the robustness of classical architectures, while the superimposed AlexNet model slightly improved upon the original in terms of accuracy.

Proposed Model with Adam Optimizer (Learning Rate: 0.001)

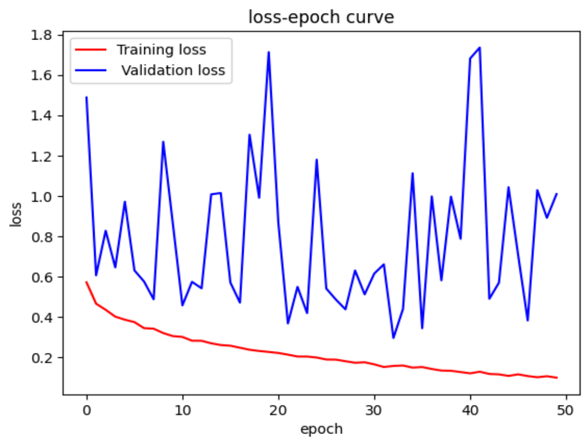


Fig 4: Loss-Epoch Curve for Proposed Model with Adam Optimizer (Learning Rate: 0.001)

The model trained using the Adam optimizer with a learning rate of 0.001 showed a steady decrease in training loss, indicating consistent learning across epochs. However, the validation loss demonstrated minor fluctuations, suggesting slight overfitting as the model may be learning noise from the training data.

The proposed model with Adam Optimizer (Learning Rate: 0.001) yielded an overall accuracy of 73%

Category	Precision	Recall	F1-Score	Support	dry_asphalt_smooth	dry_concrete_smooth	dry_gravel
dry_asphalt_smooth	0.56	0.93	0.70	2350			
dry_concrete_smooth	0.89	0.70	0.79	2350			
dry_gravel	0.98	0.55	0.70	2350			
dry_asphalt_smooth					2180	153	17
dry_concrete_smooth					688	1655	7
dry_gravel					1005	54	1291

Table 3: Classification report and confusion matrix for Proposed Model with Adam Optimizer (Learning Rate: 0.001)

Proposed Model with SGD Optimizer (Learning Rate: 0.001)

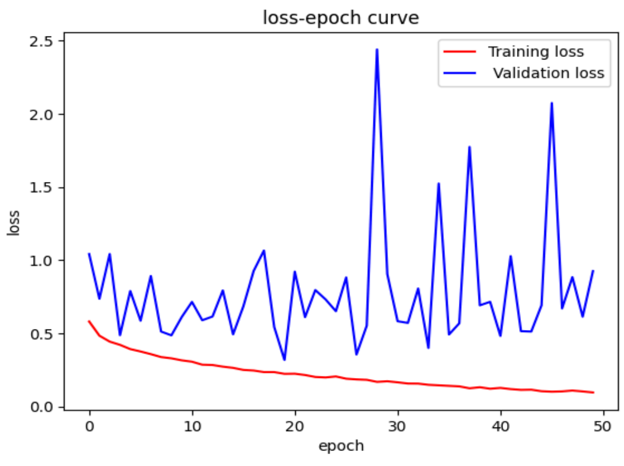


Fig 5: Loss-Epoch Curve for Proposed Model with SGD Optimizer (Learning Rate: 0.001)

Training with the SGD optimizer at a learning rate of 0.001 revealed a gradual decrease in training loss, while the validation loss displayed considerable variability. This variation could imply that the model is sensitive to the training data's nuances, which might not generalize well to unseen data.

The proposed model with SGD Optimizer (Learning Rate: 0.001) yielded an overall accuracy of 75%

Category	Precision	Recall	F1-Score	Support	dry_asphalt_smooth	dry_concrete_smooth	dry_gravel
dry_asphalt_smooth	0.91	0.31	0.46	2350			
dry_concrete_smooth	0.72	0.90	0.80	2350			
dry_gravel	0.70	0.98	0.82	2350			
dry_asphalt_smooth					718	810	822
dry_concrete_smooth					55	2116	179
dry_gravel					14	28	2308

Table 4: Classification report and confusion matrix for Proposed Model with SGD Optimizer (Learning Rate: 0.001)

Proposed Model with SGD Optimizer (Learning Rate: 0.1)

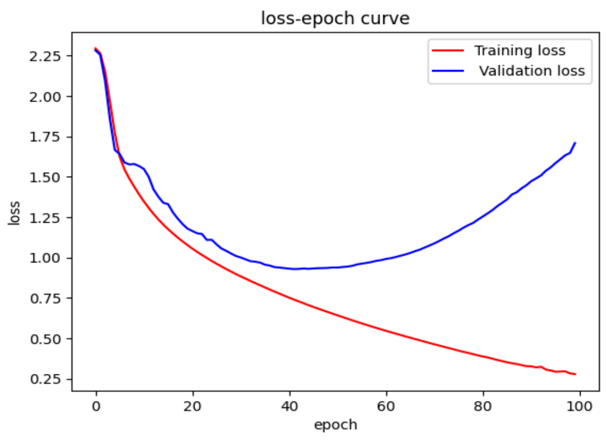


Fig 6: Loss-Epoch Curve for Proposed Model with SGD Optimizer (Learning Rate: 0.1)

When the learning rate was increased to 0.1 using the SGD optimizer, the training loss decreased sharply. The validation loss initially followed the training loss but started to increase after around 50 epochs, suggesting that the model began to overfit the training data.

The proposed model with SGD Optimizer (Learning Rate: 0.1) yielded an overall accuracy of 86%.

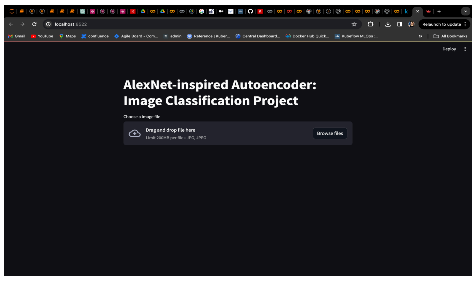
Category	Precision	Recall	F1-Score	Support	dry_asphalt_smooth	dry_concrete_smooth	dry_gravel
dry_asphalt_smooth	0.80	0.82	0.81	2350			
dry_concrete_smooth	0.87	0.83	0.85	2350			
dry_gravel	0.91	0.93	0.92	2350			
dry_asphalt_smooth					1916	241	193
dry_concrete_smooth					362	1955	33
dry_gravel					123	44	2183

Table 5: Classification report and confusion matrix for Proposed Model with SGD Optimizer (Learning Rate: 0.1)

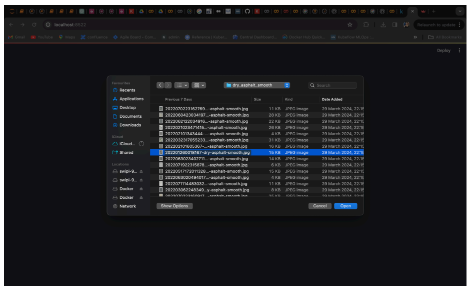
The results from the various experiments highlight the significance of the optimization algorithm and learning rate in the model's ability to correctly classify images. The SGD optimizer with a learning rate of 0.1 achieved the highest accuracy (86%), demonstrating its effectiveness for this dataset and model architecture.

Web Application

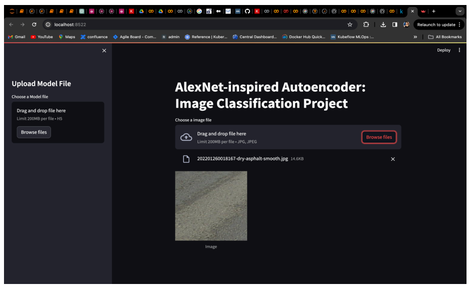
Below is the step-by-step process of the image classification task implemented by our model on our website application.


AlexNet-inspired Autoencoder: Image Classification Project
Choose a image file
Drag and drop file here
Limit 200MB per file • JPG, JPEG
Browse files
Deploy

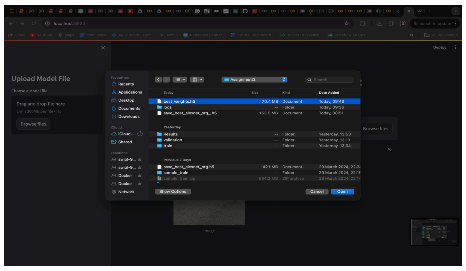
Step 1: Option for selecting a sample file



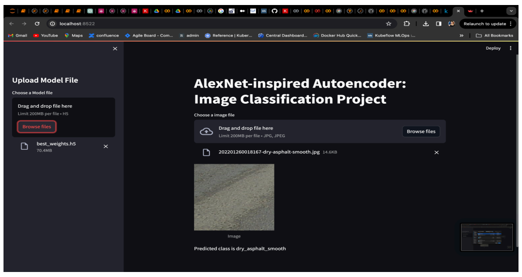
Step 2: Selecting the file



Step 3: Option for selecting the model



Step 4: Selecting the model



Upload Model File

Choose a Model file

best_weights.h5 70.4MB

AlexNet-inspired Autoencoder: Image Classification Project

Choose a image file

202201260018167-dry-asphalt-smooth.jpg 14.6KB

Predicted class is dry_asphalt_smooth

Step 5: Prediction of the selected image category by the model

VI. Conclusion

The model we developed represents a significant achievement in the realm of road surface classification using machine learning. By architecturally synthesizing the deep feature extraction of AlexNet with the reconstructive acuity of an encoder-decoder network, our approach has yielded a robust classifier. The impressive test accuracy demonstrates the model's adeptness at interpreting complex image data and accurately categorizing different road textures.

Throughout its training, we leveraged a suite of techniques, including real-time performance tracking via TensorBoard and strategic model checkpointing, changing optimizers, and learning rate to ensure the fidelity and efficacy of the learning process. The high precision, recall, and F1 scores across categories confirm the model's capacity for precise identification, a critical aspect for real-world applications.

In the future, it is important to focus on operational efficiency as much as model accuracy. Techniques like pruning and quantization can significantly reduce the model's size and computational demands without substantial loss in accuracy. Pruning removes less important connections between neurons, while quantization reduces the precision of the numbers used to represent model parameters, both leading to a lighter, faster model. Careful analysis and optimization of each layer's operations can uncover redundancies and inefficiencies. Techniques like depthwise separable convolutions, found in architectures like MobileNet, can offer substantial computational savings by decoupling the filtering and combination steps in convolutional layers. By focusing development efforts on making the model leaner and faster, without sacrificing its decision-making capabilities, we should be able to pave the way for its integration into autonomous driving systems and smart city infrastructure, marking a significant step forward in intelligent transportation technologies.

References

[1] Li, C.; Miao, F.; Gao, G. A Novel Progressive Image Classification Method Based on Hierarchical Convolutional Neural Networks. *Electronics* **2021**, *10*, 3183. https://doi.org/10.3390/electronics10243183

[2] Zhang, X.; He, L.; Qin, K.; Dang, Q.; Si, H.; Tang, X.; Jiao, L. SMD-Net: Siamese Multi-Scale Difference-Enhancement Network for Change Detection in Remote Sensing. *Remote Sens.* **2022**, *14*, 1580. https://doi.org/10.3390/rs14071580.

[3] Shengrong, Li & Changchun, Yang & Hui, Sun & Zhang, Hao. (2019). Seismic fault detection using an encoder–decoder convolutional neural network with a small training set. Journal of Geophysics and Engineering. 16. 10.1093/jge/gxy015.

[4] B. Kumeda, F. Zhang, F. Zhou, S. Hussain, A. Almasri and M. Assefa, "Classification of Road Traffic Accident Data Using Machine Learning Algorithms," 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 2019, pp. 682-687, doi: 10.1109/ICCSN.2019.8905362.

[5] M. Gogoi and S. A. Begum, "Image Classification Using Deep Autoencoders," 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Coimbatore, India, 2017, pp. 1-5, doi: 10.1109/ICCIC.2017.8524276.