Anomaly Detection Using Computer Vision

Ronok Ghosal

Table of Contents

Abstract

Anomaly detection using computer vision has gained significant importance in diverse domains, including surveillance, quality control, and cybersecurity. This paper presents a comprehensive investigation into anomaly detection using computer vision, covering principles, methodologies, applications, evaluation, and future directions. The paper highlights the challenges of detecting abnormal patterns and behaviors, emphasizing the potential of computer vision-based methods over traditional approaches. It explores feature extraction techniques, machine learning algorithms, and the role of transfer learning and generative models. Applications in video surveillance, quality control, and cybersecurity are discussed, showcasing the effectiveness of computer vision-based anomaly detection. The paper also addresses evaluation and benchmarking, including datasets and metrics. Future directions include multi-modal anomaly detection, real-time detection, and the integration of domain knowledge. This paper serves as a valuable resource for researchers and practitioners interested in utilizing computer vision for anomaly detection.

Anomaly detection, also known as outlier detection, is a fundamental task in data analysis and pattern recognition. It involves identifying instances or patterns that deviate significantly from the norm or expected behavior within a given dataset. Anomalies can arise due to various factors, such as errors in data collection, rare events, fraudulent activities, or system malfunctions. Detecting anomalies is crucial in numerous domains, including finance, healthcare, cybersecurity, manufacturing, and transportation, as anomalies often indicate critical and potentially harmful events.

In recent years, computer vision has emerged as a powerful tool for anomaly detection, leveraging the ability of machines to perceive and interpret visual data. By harnessing the capabilities of computer vision techniques, such as image and video analysis, object recognition, and motion tracking, anomaly detection can be performed with high accuracy and efficiency. Computer vision enables the extraction of meaningful information from visual data, allowing the identification of anomalies based on visual patterns and characteristics. For example, in surveillance applications, computer vision algorithms can detect suspicious behaviors, such as unauthorized access, loitering, or violent actions, by analyzing real-time video feeds. In quality control scenarios, computer vision can identify defects or abnormalities in manufacturing.

processes or product inspections, ensuring that only high-quality items are delivered to customers.

Moreover, computer vision-based anomaly detection has significant implications for cybersecurity. By analyzing network traffic or system logs, computer vision algorithms can identify anomalous patterns indicating potential cyber threats or intrusion attempts. This approach enhances the ability to detect sophisticated attacks that may otherwise go unnoticed by traditional rule-based or signature-based methods.

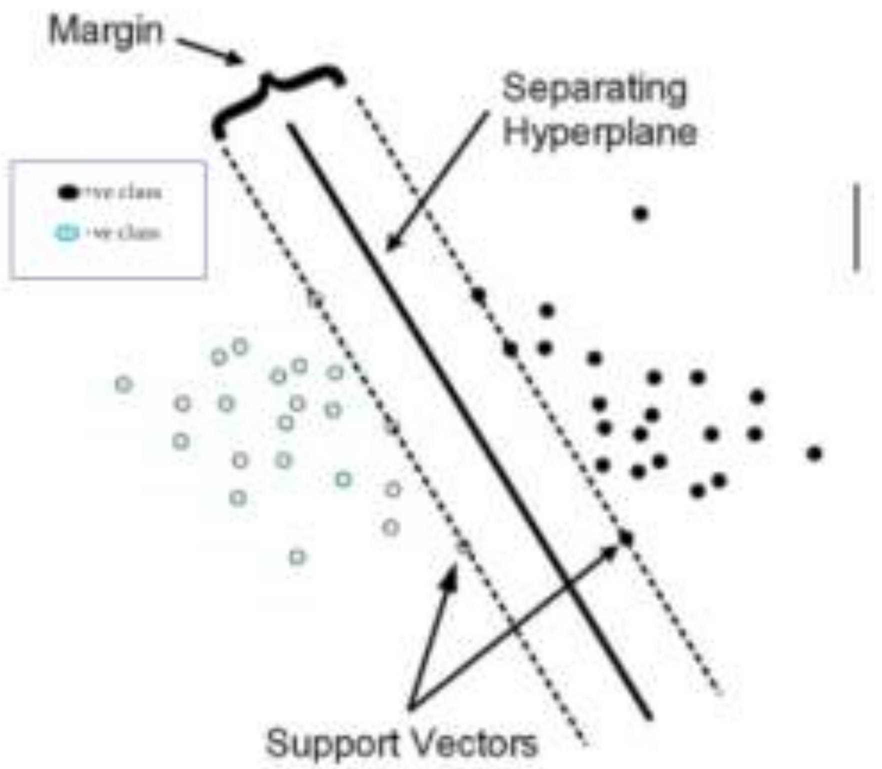
The integration of computer vision with anomaly detection opens new possibilities for automated, real-time, and accurate anomaly detection in a wide range of applications. By leveraging the power of visual perception, computer vision techniques provide valuable insights into detecting anomalies and enabling proactive decision-making to mitigate potential risks and improve overall system performance.

Keywords: Data augmentation, malaria-infected blood cell, of Convolutional Autoencoders, Kernel Density Estimation, F-1 Score, anomaly detection, Threshold Determination

Methodology

The importance of anomaly detection techniques extends to various domains. In the field of cybersecurity, anomaly detection plays a critical role in identifying malicious activities or intrusions that deviate from normal network behavior. In industrial settings, anomaly detection is employed to identify defective products, monitor equipment health, and prevent system failures. In healthcare, anomaly detection aids in the early detection of diseases, identifying anomalies in medical images or patient data that could indicate abnormalities or potential risks. Furthermore, in autonomous vehicles, anomaly detection techniques are essential for detecting unexpected events or objects that could jeopardize safety on the road. The applications of anomaly detection and computer vision extend to many domains, however studying the existing algorithms for anomaly detection in the field of computer vision is of central importance. In this paper, three major anomaly detection algorithms will be discussed.

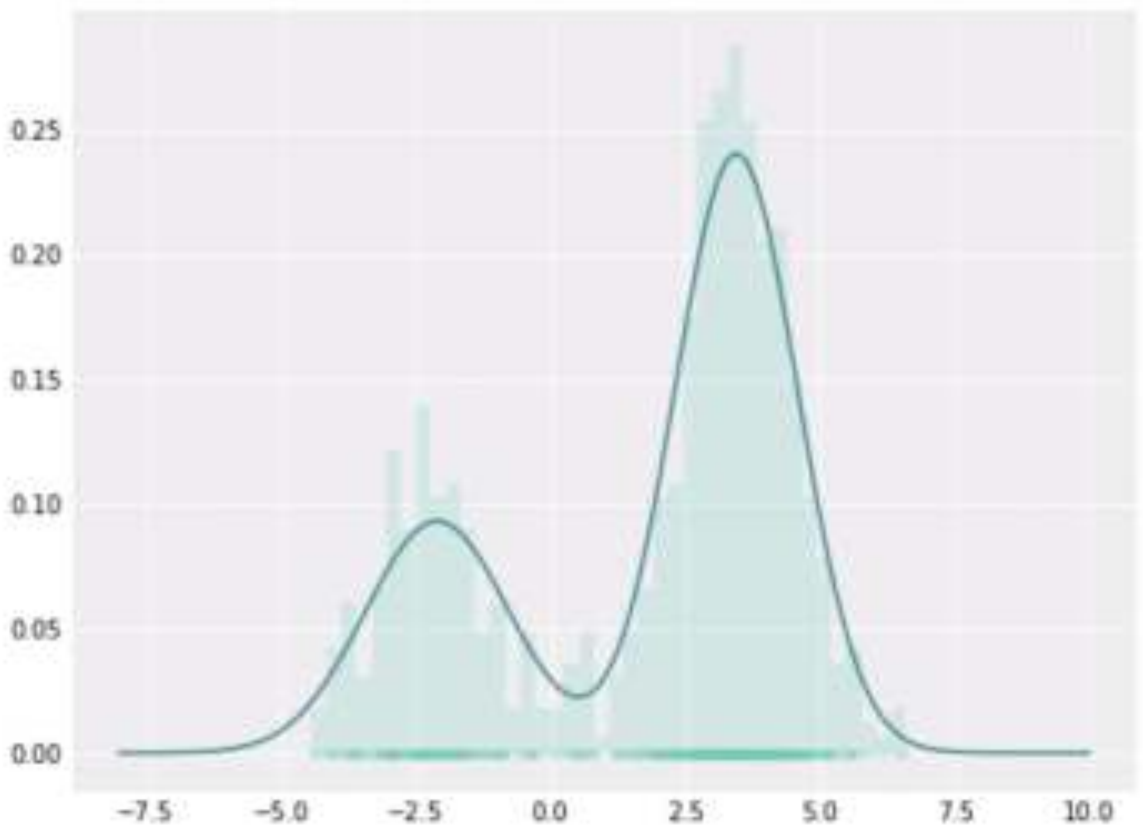
To begin the analysis of anomaly detection, the first algorithm to assess should be the on-Class Support Vector Machine or the One-Class SVM. The algorithm is trained on a set of normal or representative data, without any anomalies, to find a hyperplane that encloses most of the training data points. This hyperplane is determined by maximizing the margin, which represents the distance between the hyperplane and the closest training points.



During the testing phase, the algorithm evaluates new, unseen data points by calculating their distance from the trained hyperplane. If a test point falls within or close to the hyperplane, it is considered normal; otherwise, if it is significantly far from the hyperplane, it is classified as an anomaly. One-class SVM separates normal data from abnormal data by defining a decision boundary, assuming that normal data points are confined to a specific region in the feature space. Support vectors are the data points that lie closest to the decision boundary or hyperplane, and they play a crucial role in defining the decision boundary and making predictions. The Once Class SVM algorithm can be applied in computer vision tasks such as object detection, image segmentation, and outlier detection to recognize common patterns and structures in images and identify unusual or anomalous images that do not conform to the learned patterns.
The next algorithm to be discussed is the Gaussian Mixture Model (GMM). GMM assumes that the data is generated from a mixture of multiple Gaussian distributions. In the context of anomaly detection, GMM learns the normal data distribution by estimating the parameters of these Gaussian distributions, such as mean and covariance (the measure of how well two variables will fluctuate with each other).

During training, the algorithm uses regular instances to estimate these parameters. Anomalies can then be identified by calculating the likelihood of an instance under the learned GMM. Instances with low probabilities are considered anomalies as they deviate from the expected normal behavior.

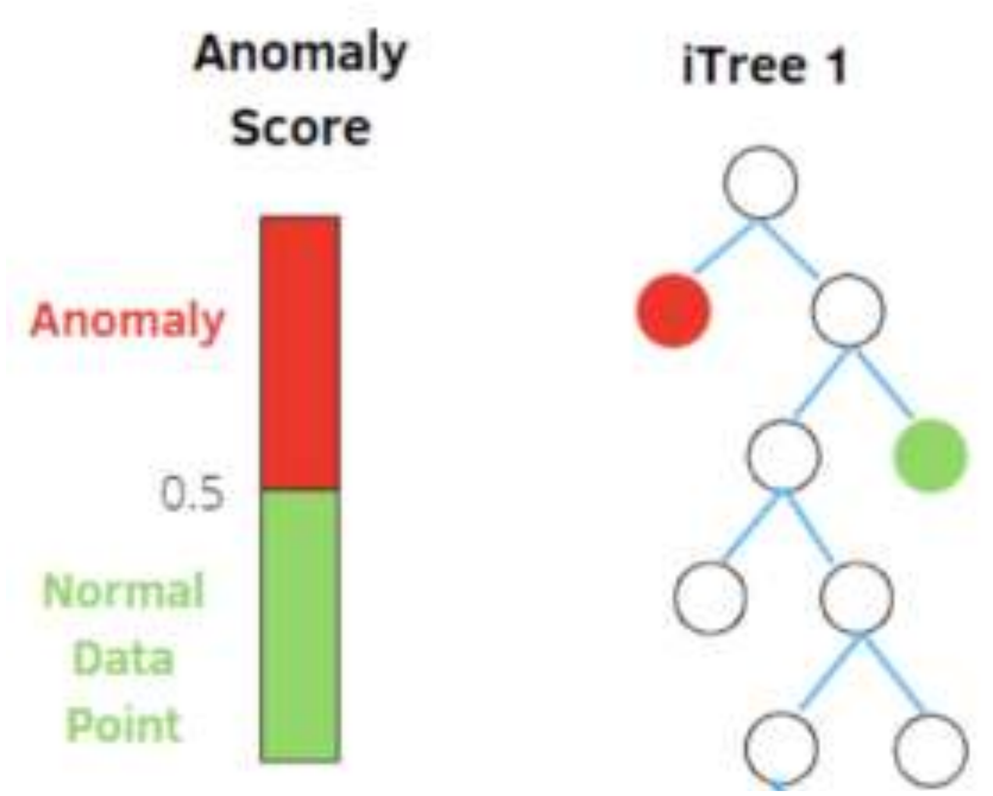
$$\text{Cov}(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$



GMM provides a robust and flexible approach for modeling complex data distributions, making them effective in detecting anomalies in computer vision applications. GMM is an algorithm widely used in anomaly detection in computer vision due to its effectiveness and efficiency in various scenarios. GMM is particularly crucial in situations where the data distribution is complex and cannot be easily represented by a single Gaussian distribution. For example, in image processing tasks such as object detection, where the visual appearance of objects can vary significantly, GMMs can accurately model the multi-modal nature of the data. GMMs are also efficient when dealing with high-dimensional feature spaces since they capture the statistical properties of the data through a set of parameters, rather than explicitly modeling each data point. This allows GMMs to efficiently represent and detect anomalies in large-scale computer vision datasets. Finally, GMMs can handle imbalanced datasets, where anomalies are rare compared to normal instances, by assigning low probabilities to outlier instances, thereby enabling effective anomaly detection in such scenarios.

The third main computer vision central anomaly detection algorithm is the isolation forest. Isolation Forest is an incredibly useful algorithm in the field of computer vision for detecting anomalies or outliers within datasets. Unlike traditional algorithms, Isolation Forest stands out due to its ability to handle high-dimensional data efficiently, making it ideal for the complex and diverse nature of computer vision applications.

What makes Isolation Forest truly special is that it doesn't rely on specific distribution assumptions for the data, making it highly versatile and adaptable to a wide range of scenarios. The algorithm itself works by employing a clever strategy: it randomly selects a feature and a random split value within the range of that feature. It then recursively partitions the data until each instance is isolated, forming individual leaf nodes. By doing so, Isolation Forest effectively captures the notion that anomalies are often few and different, causing them to be isolated more frequently within the tree structure.
To detect anomalies, Isolation Forest measures the average path length for each instance within the tree. Anomalies tend to have shorter average path lengths compared to normal instances. This allows the algorithm to assign anomaly scores, with shorter paths indicating higher degrees of anomaly.



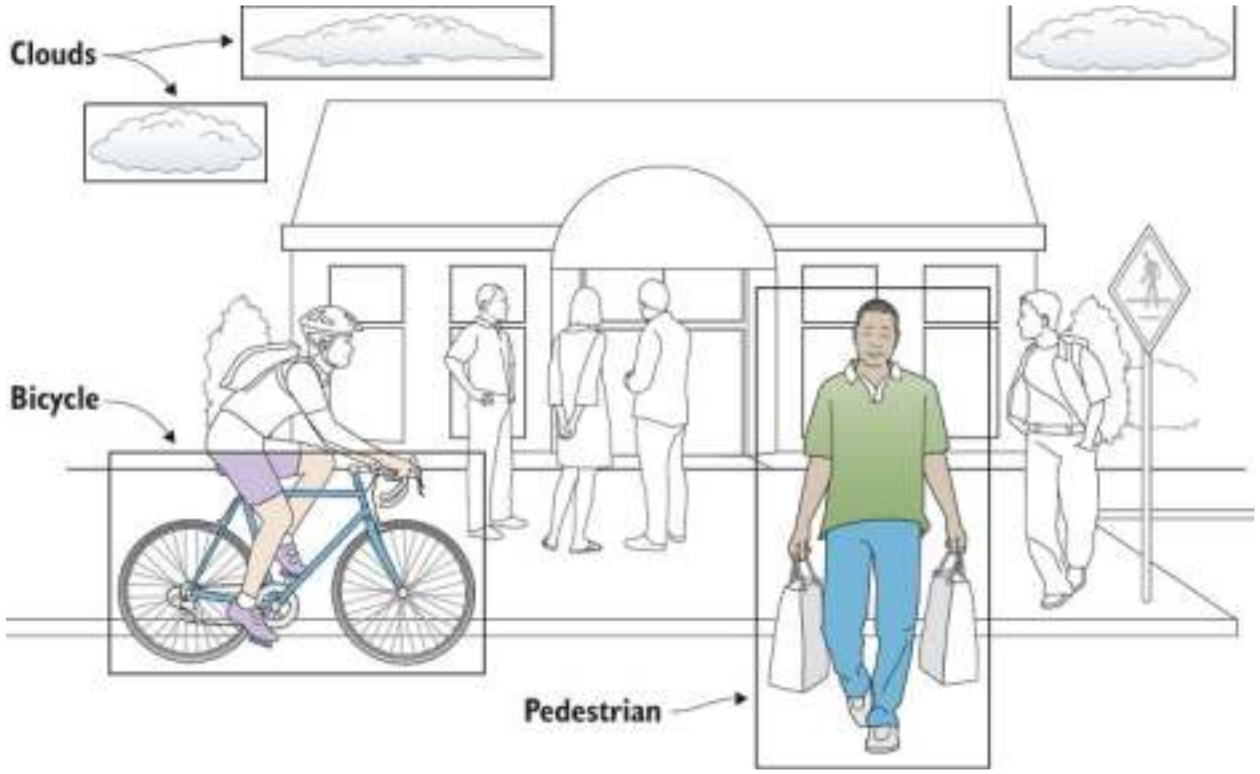
This capability makes Isolation Forest versatile, as it can detect both global and local anomalies within the dataset.
Another key advantage of Isolation Forest is its scalability. It can handle large datasets efficiently, which is crucial in computer vision applications that often deal with vast amounts of visual information. By leveraging Isolation Forest alongside other anomaly detection techniques such as Gaussian Mixture Models (GMM) and one-class Support Vector Machines (SVM), computer vision systems can benefit from a multi-faceted approach to anomaly detection, resulting in improved accuracy and robustness.

Image Preprocessing

In machine learning projects in general, you usually go through a data preprocessing or cleaning step. As a machine learning engineer, you'll spend a good amount of your time cleaning up and preparing the data before you build your learning model. The goal of this step is to make your data ready for the ML model to make it easier to analyze and process computationally, as it is with images. Based on the problem you're solving and the dataset in hand, there's some data massaging required before you feed your images to the ML model.
Image processing could be simple tasks like image resizing. To feed a dataset of images to a convolutional network, they must all be the same size. Other processing tasks can take place like geometric and color transformation, converting color to grayscale, and many more.
The acquired data are usually messy and come from different sources. To feed them to the ML model (or neural network), they need to be standardized and cleaned up. Often, preprocessing is used to conduct steps that reduce the complexity and increase the accuracy of the applied algorithm. We can't write a unique algorithm for each of the conditions in which an image is taken, thus, when we acquire an image, we tend to convert it into a form that allows a general algorithm to solve it.
Data preprocessing techniques might include:

1. Convert color images to grayscale to reduce computation complexity.

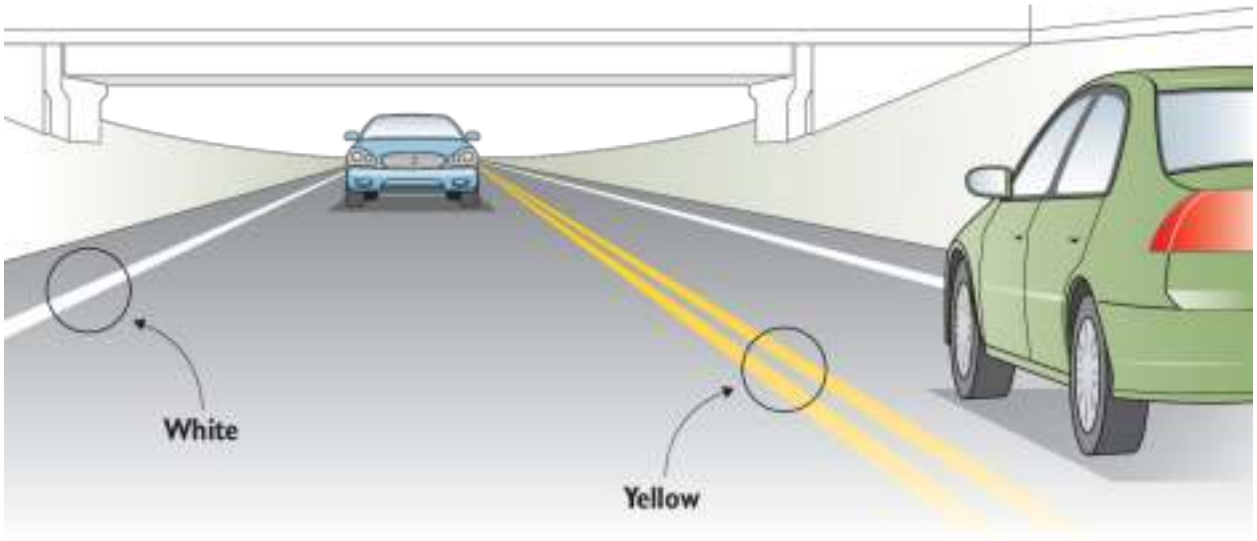
In certain problems, you'll find it helpful to lose unnecessary information from your images to reduce space or computational complexity.
For example, converting your colored images to grayscale images. This is because, in many objects, color isn't necessary to recognize and interpret an image. Grayscale can be good enough for recognizing certain objects. Because color images contain more information than black and white images, they can add unnecessary complexity and take up more space in memory (Remember how color images are represented in three channels, which means that converting it to grayscale reduces the number of pixels that need to be processed).



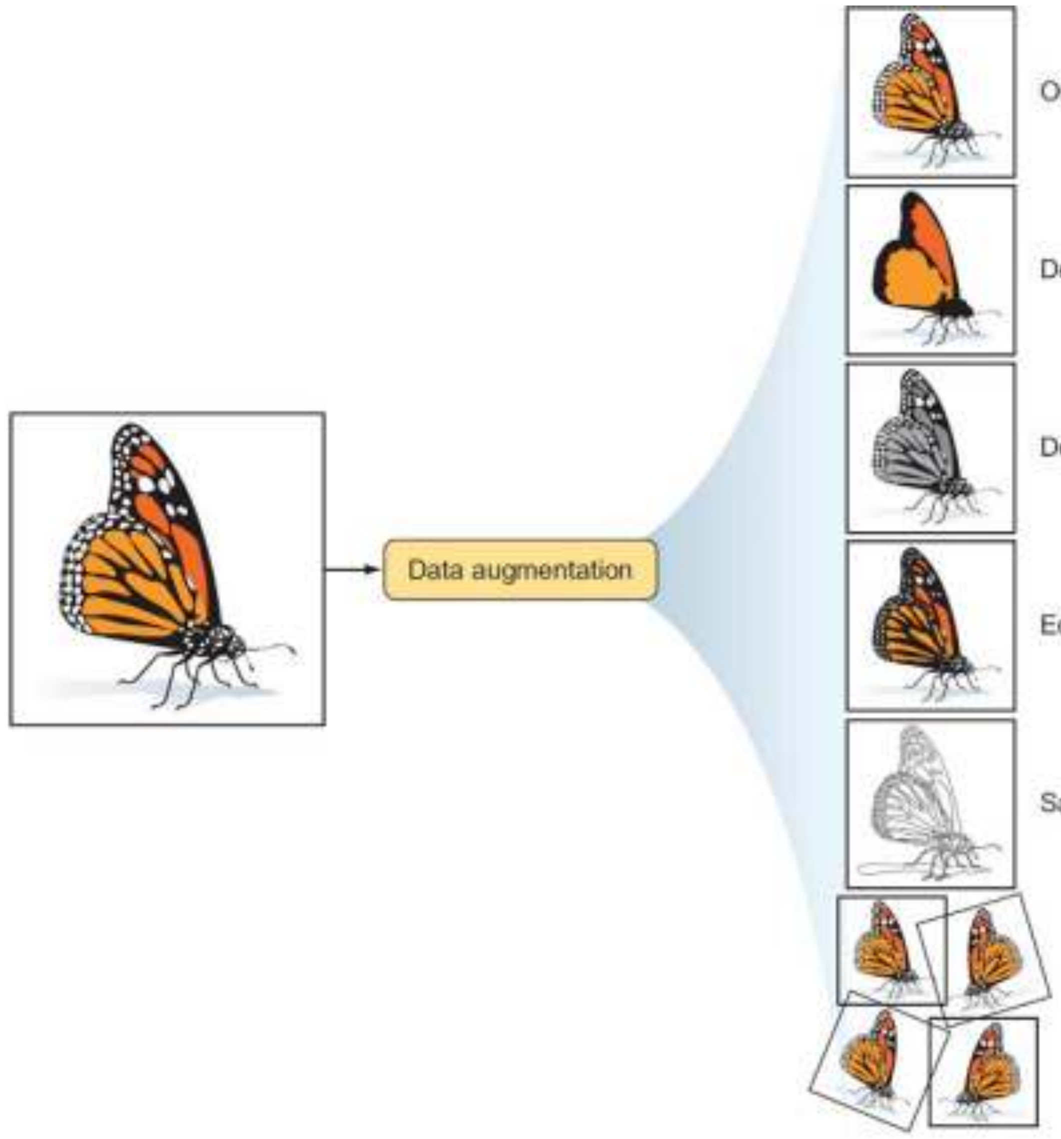
In the example above, you can see how patterns in brightness and darkness of an object (intensity) can be used to define the shape and characteristics of many objects. In other applications, color is important to define certain objects. Like skin cancer detection which relies heavily on skin colors (red rashes).

2. Color Is Important.

Converting an image to grayscale might not be a good decision for some problems. There are several applications for which color is very important. For example, building a diagnostic system to identify red skin rashes in medical images. This system relies heavily on the intensity of the red color in the skin. Removing colors from the image will make it harder to solve this problem. In general, color images provide very helpful information in many medical applications.
Another example of the importance of color in images is lane detection applications in self-driving cars. Where the car must identify the difference between yellow and white lanes because they are treated differently. Grayscale images do not provide enough information to distinguish between the yellow and white lanes.



The rule of thumb to identify the importance of colors in your problem is to look at the image with the human eye, If you can identify the object that you are looking for in a gray image then you probably have enough information to feed to your model. If
not, then you need more information (colors) in your images. The same rule can be applied to most other preprocessing techniques that will be discussed next.
3. Standardize images: One important constraint that exists in some machine learning algorithms, such as CNN, is the need to resize the images in your dataset to a unified dimension. This implies that our images must be preprocessed and scaled to have identical widths and heights before being fed to the learning algorithm.
4. Data augmentation: Another common pre-processing technique involves augmenting the existing dataset with perturbed versions of the existing images. Scaling, rotations, and other affine transformations are typical. This is done to enlarge your dataset and expose the neural network to a wide variety of variations of your images. This makes it more likely that your model recognizes objects when they appear in any form and shape. Here’s an example of image augmentation applied to a butterfly image:



5. Other techniques: Many preprocessing techniques can be used to get your images ready to train the machine learning model. In some projects, you might need to remove the background color from your images to reduce the noise. Other projects might require that you brighten or darken your images. In short, any adjustments that you need to apply to your dataset are considered a sort of preprocessing. You'll select the appropriate processing techniques based on the dataset at hand and the problem you're solving. That builds your intuition of which ones you need when working on your projects.

Use cases of Computer Vision

Project1: Emotion Detection -

Introduction to the Project

The project revolves around the fascinating field of deep learning for facial emotion detection. Its primary goal is to develop an intelligent system capable of recognizing and categorizing human emotions based on facial expressions. This project has broad applications across various industries, including human-computer interaction, market research, and mental health recognition.

1.2 Dataset Selection:

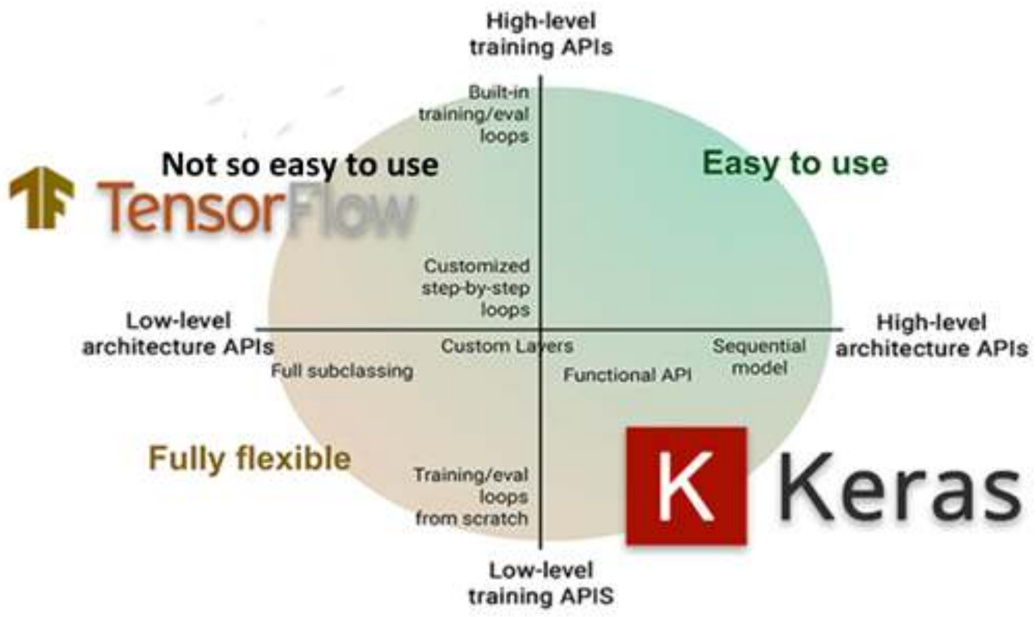
A critical component of any machine learning project is the dataset. In this case, the dataset used is Fer 2013. This dataset contains grayscale images of human faces, each associated with one of seven distinct emotions: anger, disgust, fear, happiness, neutral, sadness, and surprise.

Fer 2013 Labeled Data Sample

However, it's crucial to acknowledge that the dataset has class imbalances, with some emotions having more examples than others. Handling such imbalances is a challenge often faced in real-world applications.

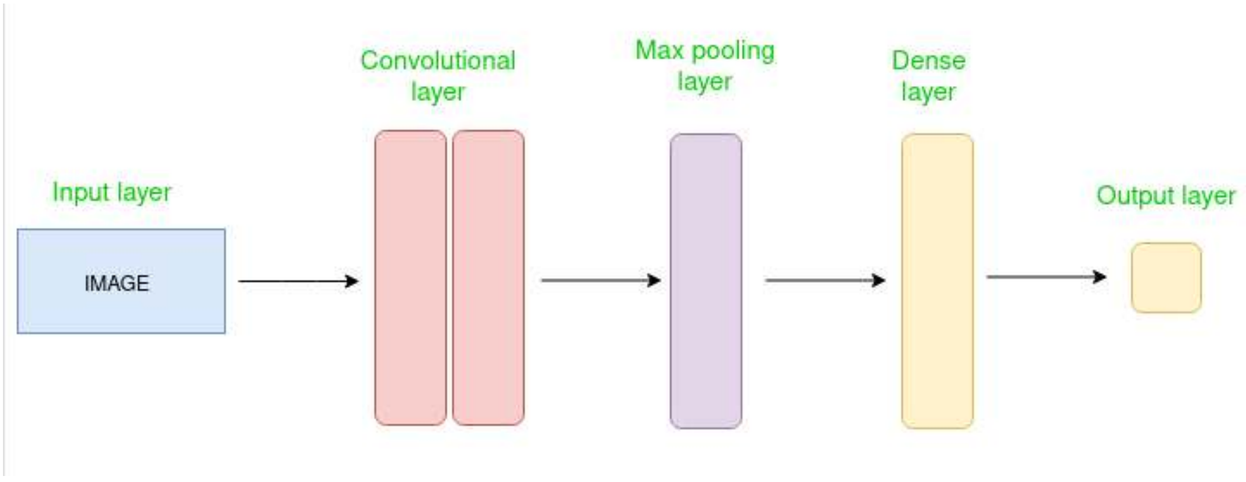
1.3. Technology Stack:

Python is the chosen programming language for this project, primarily due to its rich ecosystem of deep learning libraries. The project leans heavily on Keras, a popular high-level neural networks API that simplifies the creation and training of deep learning models.



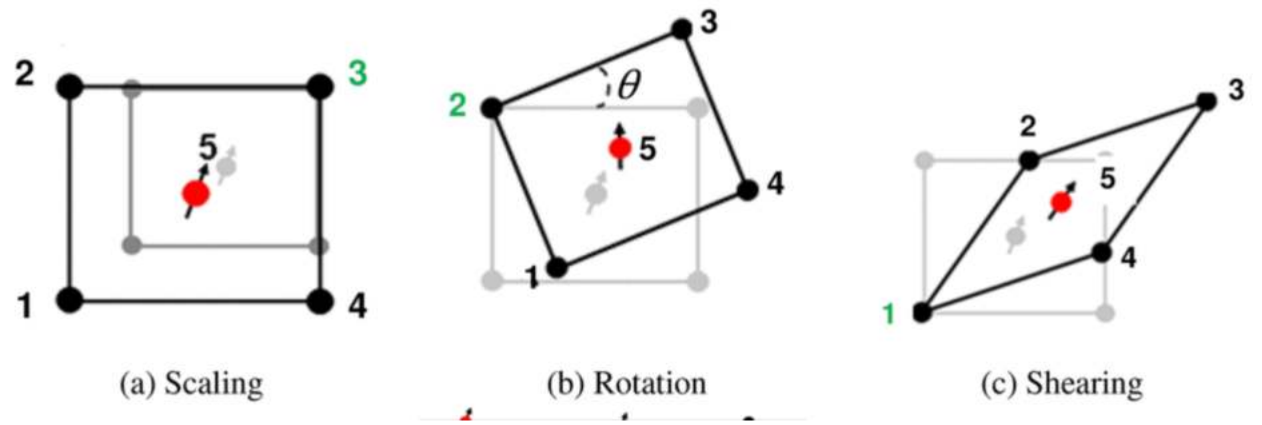
1.4. Model Architecture:

The core of the project lies in designing an effective neural network architecture. Typically, this architecture comprises convolutional layers for feature extraction. Convolutional layers are designed to recognize patterns and structures within the images. Max-pooling layers are employed for down-sampling, reducing computational complexity while retaining essential features. Fully connected layers are used for the final classification, mapping extracted features to specific emotions.



1.5. Data Preprocessing:

Before feeding the data into the neural network, several preprocessing steps are taken. The grayscale images are often resized to a consistent dimension, such as 48x48 pixels. Data augmentation techniques are applied to artificially increase the dataset's size and improve the model's generalization. Augmentation can involve operations like rotation and shearing, allowing the model to learn to recognize emotions from faces in various orientations.



Rotation Shearing

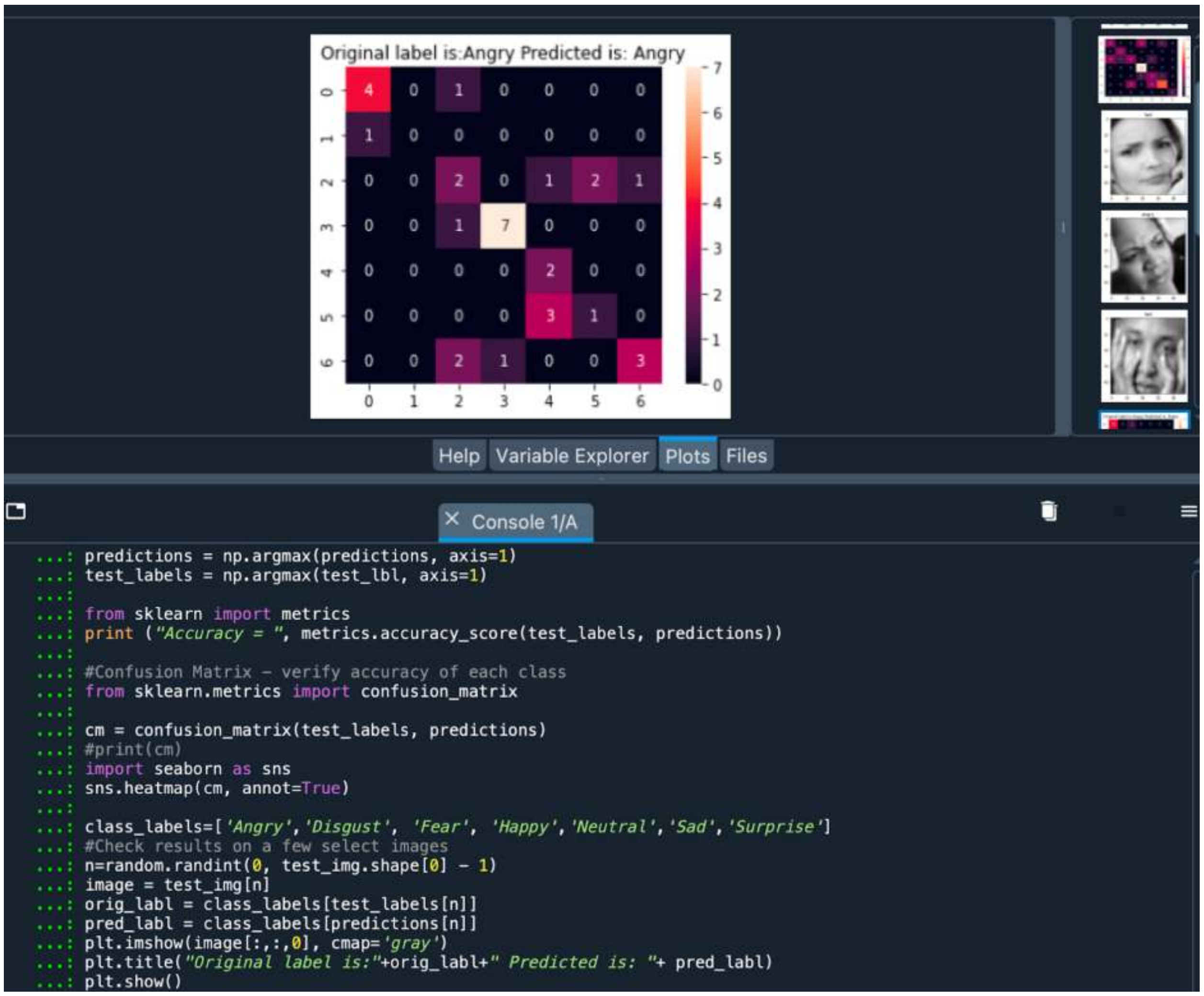
1.6. Model Training:
Training a deep learning model is an iterative process. The dataset is split into training and validation sets. The model learns from the training data and adjusts its parameters to minimize the prediction error. Validation data monitors the model's performance and prevents overfitting, a common issue in deep learning where the model becomes too specialized in the training data.

1.7. Model Evaluation:
After training, the model is evaluated using a separate test dataset. Performance metrics such as accuracy, precision, recall, and F1-score are calculated to assess the model's ability to classify emotions accurately. The confusion matrix is often used to visualize how well the model predicts each emotion and to identify areas where it may struggle.

$$\text{Accuracy} = \frac{T_P}{T_P + T_n + F_P + F_n}$$

$$\text{Recall} = \frac{T_P}{T_P + F_n}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision}_p \times \text{Recall}_p}{\text{Precision}_p + \text{Recall}_p}$$



```
...: predictions = np.argmax(predictions, axis=1)
...: test_labels = np.argmax(test_lbl, axis=1)
...:
...: from sklearn import metrics
...: print ("Accuracy = ", metrics.accuracy_score(test_labels, predictions))
...:
...: #Confusion Matrix - verify accuracy of each class
...: from sklearn.metrics import confusion_matrix
...:
...: cm = confusion_matrix(test_labels, predictions)
...: #print(cm)
...: import seaborn as sns
...: sns.heatmap(cm, annot=True)
...:
...: class_labels=['Angry','Disgust', 'Fear', 'Happy','Neutral','Sad','Surprise']
...: #Check results on a few select images
...: n=random.randint(0, test_img.shape[0] - 1)
...: image = test_img[n]
...: orig_labl = class_labels[test_labels[n]]
...: pred_labl = class_labels[predictions[n]]
...: plt.imshow(image[:,:,0], cmap='gray')
...: plt.title("Original label is:"+orig_labl+" Predicted is: "+ pred_labl)
...: plt.show()
```

1.8 Code

https://github.com/RonokGhosal/CVEmotionDetectionProject/blob/main/239_train_emotion_dete

ction/239_train_emotion_detection.py

1.8. Dataset Challenges:

The project acknowledges the challenges posed by the imbalanced dataset. Emotions like happiness and sadness may have ample examples, making them easier for the model to learn.

However, emotions like disgust might be more challenging to detect due to the limited number of training samples. This highlights the importance of having a diverse and representative dataset.

1.9. Future Developments:

The project doesn't stop at emotion detection. Future developments include expanding the scope to age and gender detection. Combining these capabilities will enable the creation of a more comprehensive facial recognition system. The goal is to integrate these models into real-time applications, such as webcam-based emotion detection, offering valuable insights into various fields.

1.10. Observation:

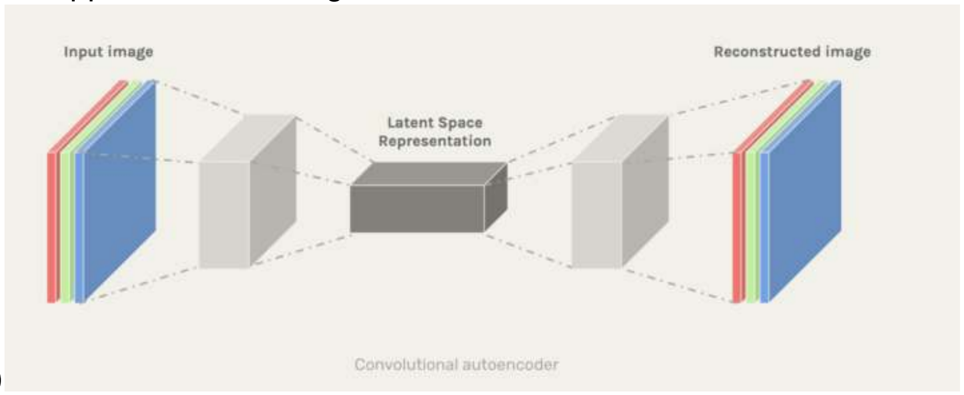
In conclusion, this project provides a deep dive into the intricate world of deep learning for facial emotion detection. It demonstrates the significance of dataset selection, model architecture, data preprocessing, training, and evaluation. It also highlights the real-world challenges posed by imbalanced datasets and sets the stage for exciting future developments in the realm of facial recognition technology.

Project 2: Anomaly detection in medical images

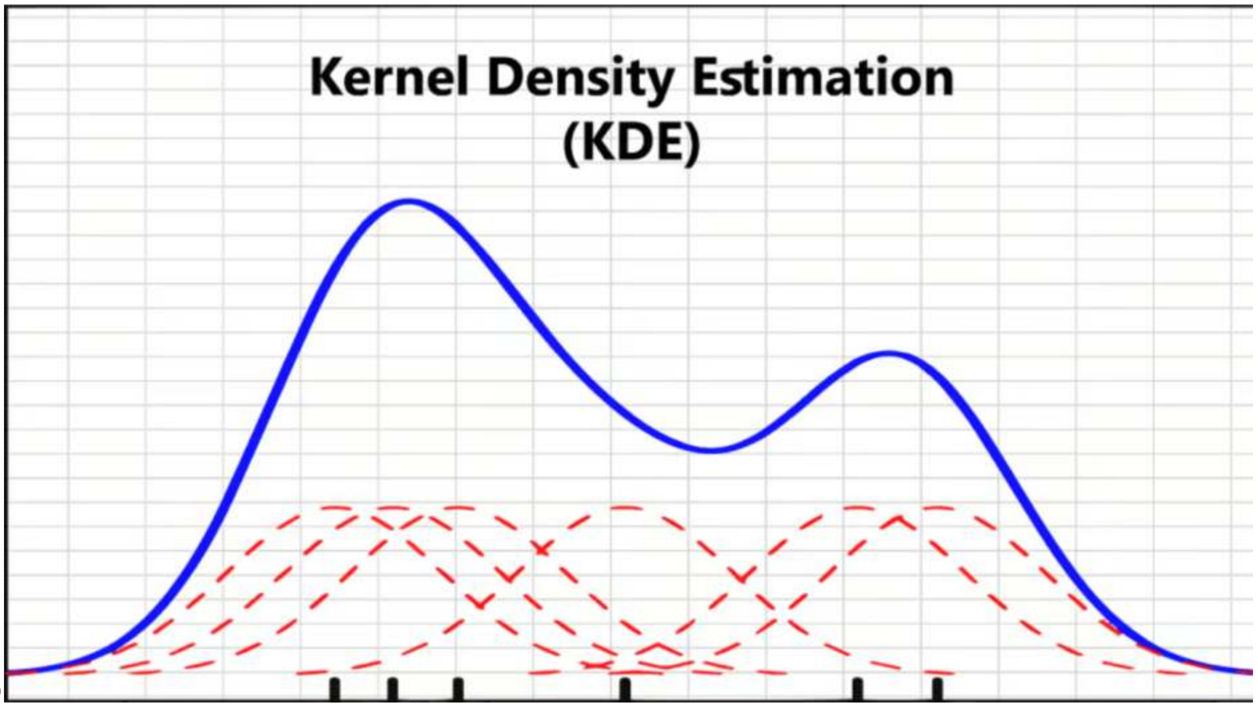
This project presents an extensive investigation into the application of Convolutional Autoencoders (CAEs) and Kernel Density Estimation (KDE) for anomaly detection in medical images. The significance of accurately identifying anomalies in medical images cannot be overstated, as it plays a pivotal role in early disease diagnosis and healthcare decision-making. In this study, we offer a comprehensive overview of the methods employed, the experimentation process, and the key findings.

2.1 Introduction:

Anomaly detection holds paramount importance in a myriad of domains, particularly in the realm of medical imaging. Detecting anomalies in medical images, such as X-rays or microscopy images, is critical for accurate diagnosis and timely treatment. This research delves into an innovative approach that amalgamates Convolutional Autoencoders



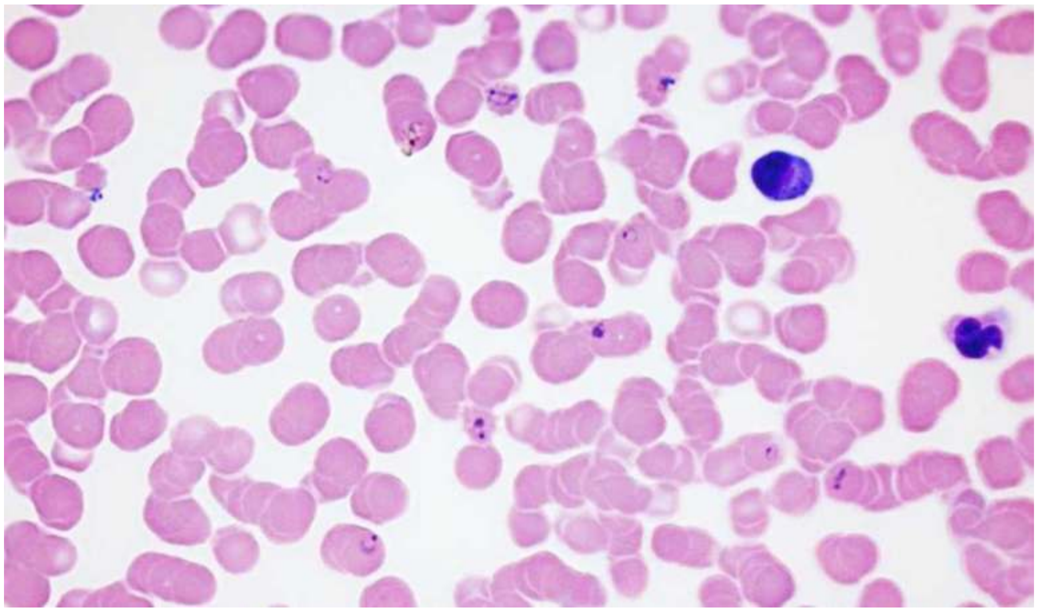
(CAEs) and Kernel Density Estimation



(KDE) to accomplish the task of anomaly detection in medical images.

2.2. Data Preparation:

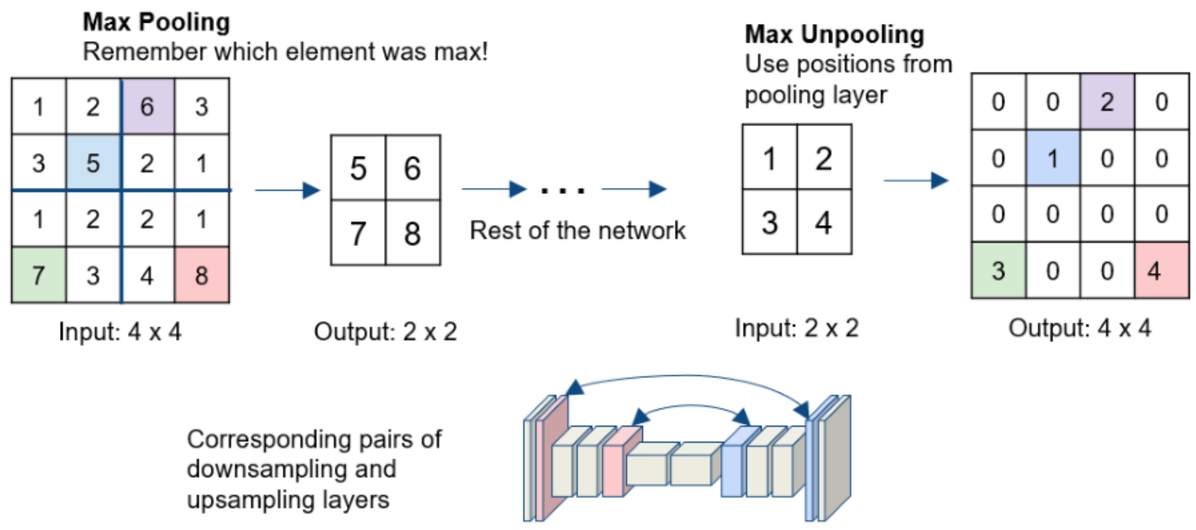
The project's inception involves data acquisition and meticulous preprocessing. A dataset containing both normal and anomaly-labeled medical images is assembled. To exemplify the capabilities of our approach, we focus on detecting anomalies in malaria-infected blood cell. images.



Data augmentation techniques are judiciously applied to augment the dataset, and image standardization is executed, resizing all images to a uniform dimension of 128x128 pixels.

Convolutional Autoencoders (CAEs):

Convolutional Autoencoders (CAEs) represent a potent neural network architecture frequently harnessed for tasks involving image reconstruction and feature extraction. The architecture of our CAE comprises convolutional layers, max-pooling, and up-sampling



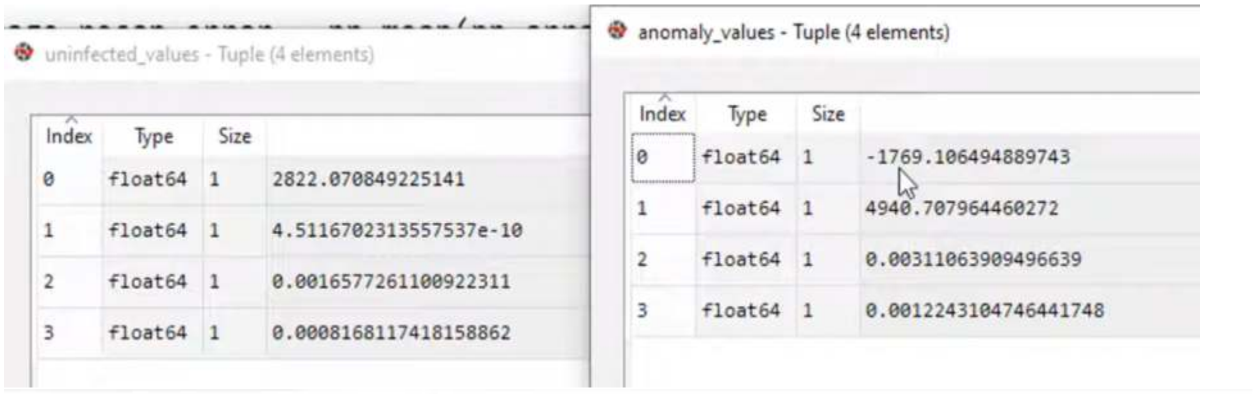
layers.
The encoder part reduces image dimensions to a bottleneck layer, while the decoder part adeptly reconstructs the image from this bottleneck representation.

2.3 Kernel Density Estimation (KDE):

Kernel Density Estimation (KDE) is a statistical technique used to estimate the probability density function of a dataset. In our study, we exploit KDE to calculate the likelihood of an image belonging to the normal class in the latent space. The latent space is derived by extracting the bottleneck layer's output from the CAE, which serves as a compact feature vector for each image.

2.4 Threshold Determination:

Effectively distinguishing anomalies from normal images necessitates the establishment of two critical threshold values: one for the density estimate in the latent space and another for the reconstruction error.
These thresholds are meticulously derived through an in-depth analysis of the statistical distribution of these metrics on the training dataset, thus ensuring the precision of anomaly detection while minimizing false positives.

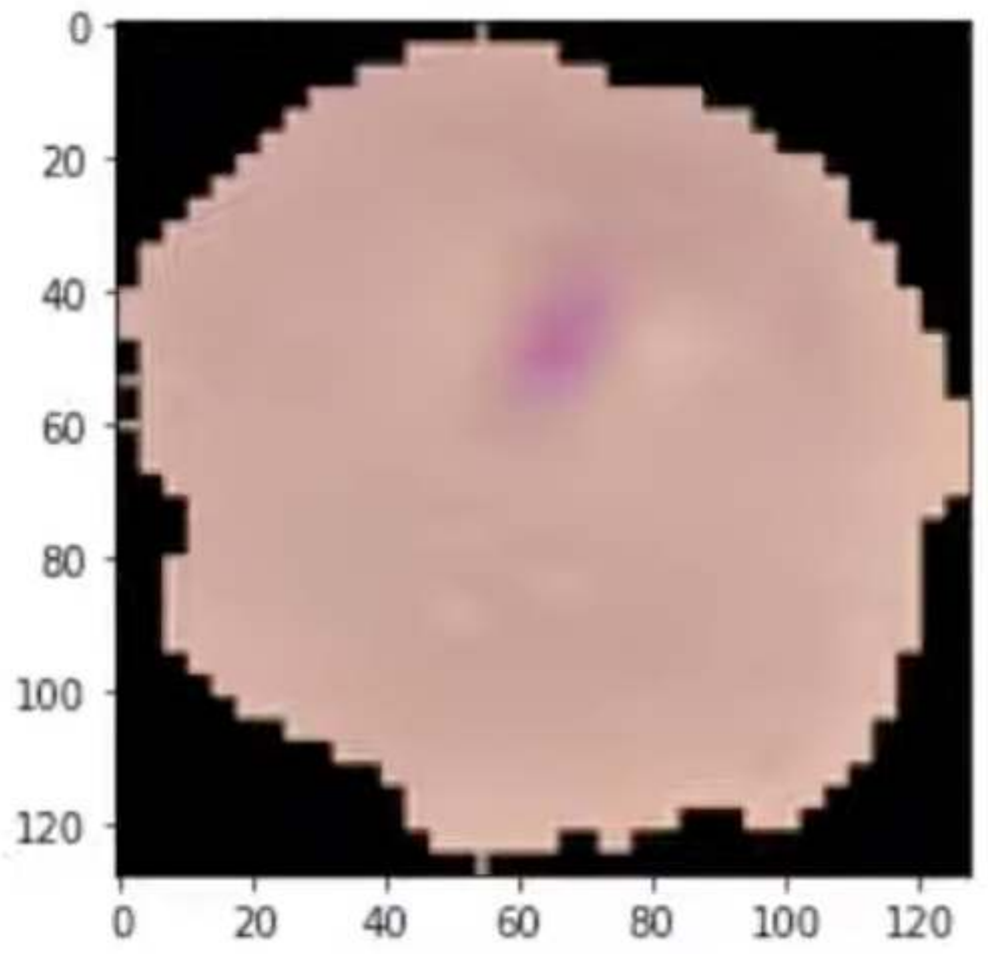


uninfected_values - Tuple (4 elements)

Index	Type	Size	
0	float64	1	2822.070849225141
1	float64	1	4.5116702313557537e-10
2	float64	1	0.0016577261100922311
3	float64	1	0.0008168117418158862

anomaly_values - Tuple (4 elements)

Index	Type	Size	
0	float64	1	-1769.106494889743
1	float64	1	4940.707964460272
2	float64	1	0.00311063909496639
3	float64	1	0.0012243104746441748



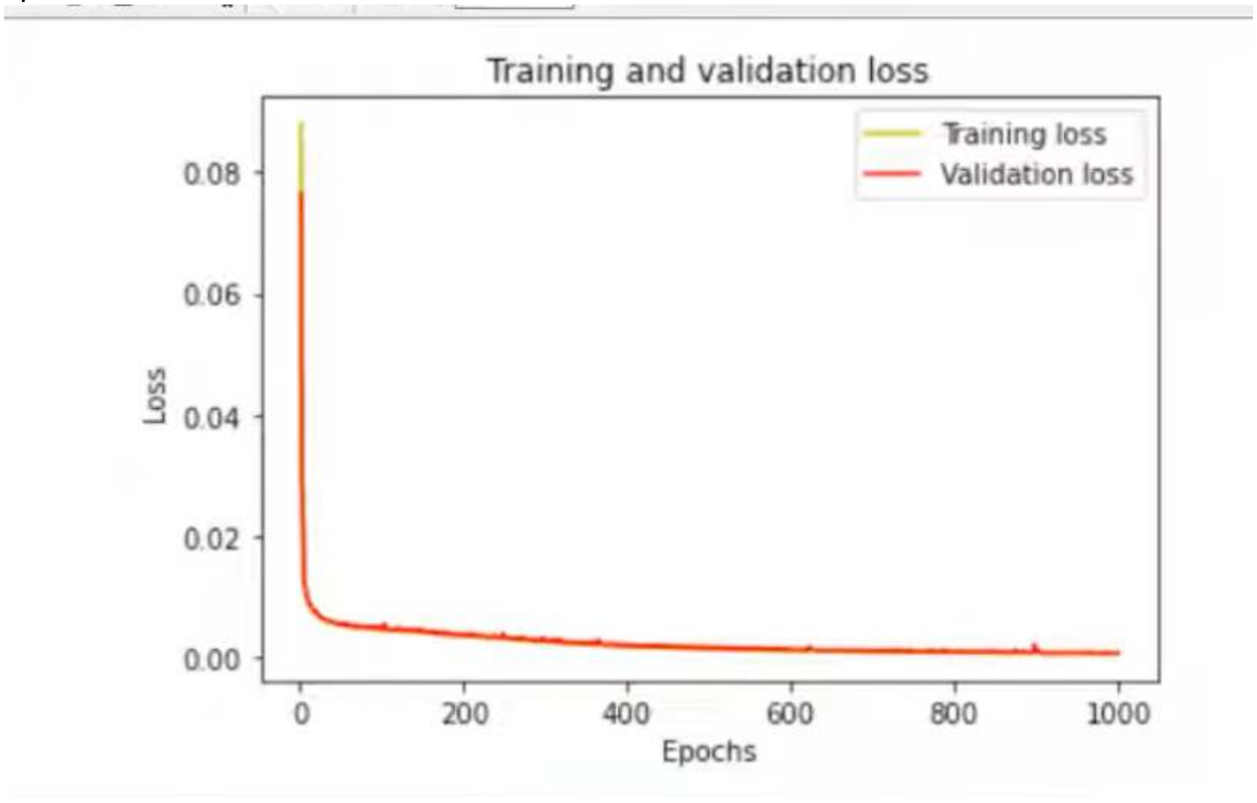
True Anomaly

2.5. Experiment:

To rigorously evaluate our proposed anomaly detection system, we meticulously curate a dataset encompassing 10,000 malaria-infected blood cell images and an equal number of normal blood cell images. This dataset's choice stems from its suitability for our research due to the evident presence of anomalies (malaria parasites) in a substantial subset of the images. A thorough data preprocessing pipeline entails resizing all images to a standardized dimension of 128x128 pixels, normalizing pixel intensities to the range [0, 1], and introducing data augmentation techniques such as random rotations and flips.

2.6 Model Training:

The training process commences with the training of the CAE architecture on 80% of the dataset, with the remaining 20% preserved for subsequent validation. A batch size of 32 is selected, and model optimization is realized through the Adam optimizer with a learning rate set at 0.001. The training is protracted, spanning 1000 epochs to ensure the model converges to an optimal state.

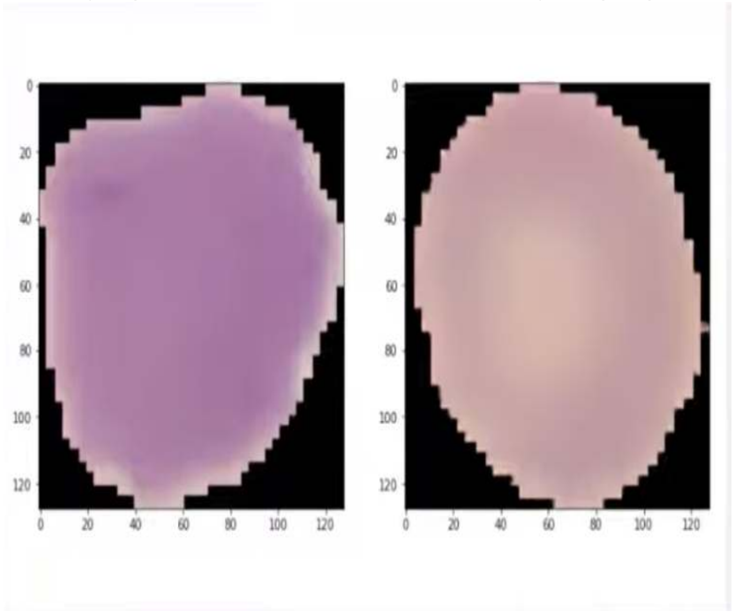


2.7 Anomaly Detection:

Following the completion of CAE training, we shift our focus to the validation dataset, which encompasses both normal and malaria-infected blood cell images. Anomaly detection unfolds through the extraction of latent space representations from the bottleneck layer of the CAE for every image. Subsequently, we compute both the KDE-based density estimate and the reconstruction error for each image, which collectively constitute the foundation for anomaly detection.

2.8 Threshold:

The ability of our anomaly detection system to accurately identify anomalies while minimizing false alarms hinges on the judicious selection of threshold values. To accomplish this, an exhaustive analysis of these metrics on the training dataset is conducted, with the objective of identifying the thresholds that optimally segregate normal images from anomalies.



2.9. Results:

2.91 Anomaly Detection:

Anomaly detection is carried out on the validation dataset, a critical step for evaluating the system's real-world performance. The trained CAE is employed to extract latent space representations, and subsequently, density estimates, and reconstruction errors are computed for each image.

2.10 Threshold Evaluation:

The thresholds established for density estimates and reconstruction errors are underpinned by a thorough statistical analysis of the training dataset. These thresholds are instrumental in effectually discerning anomalies from normal images, thus underlining the system's precision in anomaly detection while mitigating the occurrence of false positives.

4.3 Evaluation Metrics:
The performance of our anomaly detection system is rigorously evaluated through a battery of well-established metrics, including precision, recall, and F1-score. These metrics furnish a quantitative insight into the system's efficacy in accurately detecting anomalies while sustaining a low false alarm rate.

2.11. Model Robustness:

Our research endeavors to assess the robustness of the anomaly detection system through systematic experimentation, encompassing variations in model architectures, latent space dimensions, and threshold values. This rigorous testing ensures that the system maintains its effectiveness across diverse scenarios and underlines its adaptability.

2.12. Limitations:

While the fusion of CAEs and KDE yields an effective anomaly detection system, the system's performance is contingent on the quality and diversity of the training dataset. Additionally, optimal hyperparameter tuning plays a pivotal role in achieving exceptional results.

2.13. Conclusion:

In summation, this research endeavor represents a substantial stride forward in the domain of anomaly detection in medical images. The fusion of Convolutional Autoencoders and Kernel Density Estimation, as elucidated in this paper, has demonstrated exceptional promise in accurately identifying anomalies while preserving a low false-positive rate. The implications of this work extend to the augmentation of medical diagnostics and the enhancement of patient care.

2.14. Future Work

Future investigations in this domain are poised to broaden the horizons of anomaly detection, extending its purview to encompass other medical imaging modalities, and delving into advanced deep learning architectures. Additionally, the integration of clinical validation and real-world testing holds the potential to corroborate the system's effectiveness in practical healthcare settings.

2.15. Code:

https://github.com/RonokGhosal/CVEmotionDetectionProject/blob/main/260_image_anomaly_detection_using_autoencoders/260_image_anomaly_detection_using_autoencoders.py

2.16. Conclusion- Overall Takeaway:

Computer vision has made significant strides in recent years, enabling machines to perceive and understand visual information. This field relies on the use of deep learning models, such as convolutional neural networks (CNNs), to analyze and interpret images or video streams. As a result, computer vision has found diverse applications across various domains, including autonomous vehicles, surveillance systems, medical imaging, and facial recognition. Anomaly detection, a subfield of computer vision, focuses on identifying rare or unusual patterns in data. Its importance lies in detecting anomalies or outliers in images, videos, or datasets. While traditional methods relied on statistical techniques and handcrafted features, the emergence of deep learning approaches has revolutionized anomaly detection. These deep learning techniques have the advantage of automatically learning relevant features and patterns. Looking toward the future, the prospects for computer vision and anomaly detection appear promising. Ongoing advancements in hardware, such as the development of powerful graphics processing units (GPUs) and specialized chips, continue to accelerate the processing speed and efficiency of computer vision algorithms. This, in turn, facilitates real-time and edge-based applications, making computer vision more accessible and pervasive. In addition to hardware improvements, research in emerging areas holds great potential for enhancing computer vision capabilities. For instance, the advent of generative adversarial networks (GANs) has enabled realistic data synthesis, which can augment training datasets and address data scarcity challenges. Furthermore, techniques like one-shot learning and few-shot learning are being explored, allowing models to learn from limited labeled data and reducing the need for large, annotated datasets.

References

[1] Duong, Huu-Thanh, et al. “Deep Learning-Based Anomaly Detection in Video Surveillance: A Survey.” Sensors (Basel, Switzerland), U.S. National Library of Medicine, 24 May 2023, www.ncbi.nlm.nih.gov/pmc/articles/PMC10255829/. Accessed 05 July 2023.

[2] Feng, Xin & Jiang, Youni & Yang, Xuejiao & Du, Ming & Li, Xin. (2019). Computer vision algorithms and hardware implementations: A survey. Integration. 69. 10.1016/j.vlsi.2019.07.005.

[3] Khan, Ashural, et al. “Machine Learning in Computer Vision.” Procedia Computer Science, Elsevier, 16 Apr. 2020,

[4] www.sciencedirect.com/science/article/pii/S1877050920308218. Accessed 05 July 2023.

[5] Reynolds, Douglas. “Gaussian Mixture Models - Leap Laboratory.” Gaussian Mixture Models, leap.ee.iisc.ac.in/sriram/teaching/MLSP_16/refs/GMM_Tutorial_Reynolds.pdf. Accessed 05 July 2023.

[6] Salehzadeh Nobari, Amin Ebrahim, and M H Ferri Aliabadi. “A Multilevel Isolation Forrest and Convolutional Neural Network Algorithm for Impact Characterization on Composite Structures.” Sensors (Basel, Switzerland), U.S. National Library of Medicine, 19 Oct. 2020, www.ncbi.nlm.nih.gov/pmc/articles/PMC7589093/. Accessed 05 July 2023.

[7] Seliya,
Naeem, et al. “A Literature Review on One-Class Classification and Its Potential Applications in Big Data - Journal of Big Data.” SpringerOpen, Springer International Publishing, 10 Sept. 2021, journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00514-x. Accessed 05 July 2023.

[8]Takezoe, Rinyoichi, et al. “Deep Active Learning for Computer Vision: Past and Future.” arXiv.Org, 24 Dec. 2022, arxiv.org/abs/2211.14819. Accessed 05 July 2023.
Deep Learning for Vision Systems by Mohamed Elgendy
Bhattiprolu, S. (2023, August 23). 330 - Fine-tuning Detectron2 for instance segmentation using custom data