Cnn-trans model: A parallel dual-branch network for fundus image classification

**ABSTRACT**

The existence of fundus diseases not only endangers people’s vision, but also brings serious economic burden to the society. Fundus images are an objective and standard basis for the diagnosis of fundus diseases. With the continuous advancement of computer science, deep learning methods dominated by convolutional neural networks (CNN) have been widely used in fundus image classification. However, the current CNN-based fundus image classification research still has a lot of room for improvement: CNN cannot effectively avoid the interference of repeated background information and has limited ability to model the whole world. In response to the above findings, this paper proposes the CNN-Trans model. The CNN-Trans model is a parallel dual-branch network, which is the two branches of CNN-LSTM and Vision Transform (ViT). The CNN-LSTM branch uses Xception after transfer learning. As the original feature extractor, LSTM is responsible for dealing with the gradient disappearance problem in neural network iterations before the classification head, and then introduces a new type of lightweight attention mechanism between Xception and LSTM: Coordinate Attention, so as to emphasize the key information related to classification and suppress the less useful repeated background information; while the self-attention mechanism in the ViT branch is not limited by local interactions, it can establish long-distance dependence on the target and extract global features. Finally, the concatenation (Concat) operation is used to fuse the features of the two branches. The local features extracted by the CNN-LSTM branch and the global features extracted by the ViT branch form complementary advantages. After feature fusion, more comprehensive image feature information is sent to the to the classification layer. Finally, after a large number of experimental tests and comparisons, the results show that: the CNN-Trans model achieved an accuracy of 80.68% on the fundus image classification task, and the CNN-Trans model has a classification that is comparable to the state-of-the-art methods. performance.

**Keywords:** Fundus image classification Parallel dual branch Attention mechanism Feature fusion CNN.

**CHAPTER 1**

**Introduction**

With the continuous development of modern lifestyle, computers, mobile phones and other digital screens have become more and more popular. However, staring at electronic screens for a long time can cause eye fatigue, dryness and discomfort. This phenomenon is very common. The main reason is that we When using electronic devices, the frequency of blinking decreases and the eyes are not adequately lubricated. In addition, factors such as light radiation, unhygienic eye use, age decline, etc. can also cause eye damage. For individuals, eye health problems can lead to less efficient learning, as poor vision can affect work performance and academic performance. For society, people with vision problems may need more frequent eye treatments and glasses, which increases the demand for medical resources and is a significant economic burden. There are many kinds of fundus diseases, which seriously affect human visual health and are one of the main causes of blindness worldwide. Cataracts, diabetic retinopathy, age-related macular degeneration and high myopia are all common blinding diseases. Although there are many types of fundus diseases, many fundus diseases will produce certain signs in the fundus. However, since most patients have low awareness of these signs, they may ignore eye changes and delay treatment. Therefore, early detection of fundus diseases is of great significance for subsequent treatment. Fundus examination uses a professional fundus camera or scanning instrument to capture fundus images for analysis. These images show the structures inside the fundus of the eye, including the retina, optic nerve, blood vessels, and other important tissues. By observing fundus images, doctors can detect whether the fundus blood vessels are normal and whether there are abnormal expansions, lesions, or bleeding. Fundus imaging has many advantages and is a non-invasive method that does not require any surgery or irritating procedures. Secondly, fundus images provide objective visual information, reducing the impact of subjective factors on diagnosis. Doctors can assess the status of systemic blood circulation, thereby early detecting and managing systemic health-related problems. Therefore, fundus examination can not only be used to detect eye diseases, such as vitreous, retina, choroid and optic nerve diseases, but also a monitoring window for many systemic diseases [1]. However, clinical diagnosis of fundus diseases relies on professional ophthalmologists. As the number of patients continues to increase, fundus image data is also becoming larger and larger. The diagnosis of fundus data will consume a lot of energy and time of ophthalmologists. And limited professional medical resources should be used in more valuable directions. In order to assist ophthalmologists to make accurate diagnosis based on fundus images, it is necessary to develop a computer-aided diagnosis system (CAD for short) for fundus image classification as soon as possible. At present, in the field of computer vision, the challenges of fundus image classification are mainly reflected in the following aspects [2,3]:

(1) Publicly available fundus image datasets are generally small in size. There are two main reasons: a. Fundus images contain sensitive personal information, such as retinal structure, etc. When processing and sharing fundus image data, appropriate privacy protection measures must be taken to ensure that patient privacy and data security are not violated. To protect patient privacy, fundus image data collected by hospitals are usually kept confidential. b. In order to accurately classify fundus images, manual annotation and verification of the data set is the first step, but this requires highly specialized knowledge and experience to ensure the accuracy of diagnostic results, therefore, mobilizing a large number of professional ophthalmologists Participating in the creation and annotation process of data sets is a difficult problem for researchers.

(2)Most fundus image datasets suffer from severe class imbalance problem, which becomes more prominent when the number of classes increases. The direct reason why category imbalance occurs is that the probability of developing fundus diseases is quite different. However, there are various factors that affect the probability of developing fundus diseases[4–6], such as: a. Geographical factors: The incidence of trachoma is usually higher in warm, dry and poor areas, which are more likely to become hotbeds for the spread of infectious diseases. For example, sub-Saharan Africa, parts of Asia, and some countries in Oceania have long been highincidence areas for trachoma. b. Hygiene factors. In areas with poor sanitary conditions, such as places lacking clean drinking water and sanitation facilities, people are more likely to be exposed to sources of infection, thereby increasing the risk of infection from bacterial or viral fundus diseases. c. Genetic factors, there are racial differences in the occurrence and development of myopia. Studies on the incidence of myopia in different races have found that in rapidly developing economies in East Asia such as China, its prevalence is growing at an alarming rate. d. Age factor. Cataracts are common in middle-aged and elderly people over 50 years old. This is because the organs in the body gradually age and the lens will also undergo degenerative changes, such as turbidity.

(3) Different models of equipment used to obtain fundus images in hospitals may have differences in image resolution, lighting conditions, imaging quality, etc.. For example, some images may be produced by old equipment that has not been updated in time, and there are cases of low pixels and resolutions. It may also be taken in a dimly lit environment, resulting in uneven brightness and darkness in various areas of the fundus image. The low-quality images caused by these factors greatly increase the difficulty of classification. In recent years, with the rapid development of deep learning technology in the field of computer vision, more and more researchers have begun to use deep learning algorithms to analyze and process fundus images. Deep learning is a machine learning method based on multilayer neural networks that can automatically learn and extract features from large amounts of data to achieve highly accurate image analysis and recognition. In fundus image analysis, deep learning technology can extract features in fundus images by training a large amount of fundus image data, and perform tasks such as classification, positioning and segmentation. In 2019, Imran Qureshi et al. [7] compared related CAD systems based on statistical parameters for quantitative evaluation. The comparison results showed that accurate development of CAD systems is still needed to assist clinical diagnosis of diabetic retinopathy.

In 2021, Chea et al. [8] found that existing computer-aided systems cannot simultaneously detect multiple major eye diseases. To better understand the multi-category classification of fundus images, they used an optimal residual deep neural processing technology. In 2023, Wen Jingyi et al. [9] believed that the symptoms of chronic kidney disease, hypertension, type 2 diabetes and other diseases could be found based on the specific performance of retinal images, and pointed out that the latest development of AI technology is the use of retinal images to diagnose kidney diseases. Rapid large-scale screening and prognosis prediction bring great potential. These research results will help establish an accurate computer-aided diagnosis (CAD) system to provide doctors with more accurate diagnosis results and treatment suggestions, thereby improving patients’ diagnosis and treatment experience and treatment effects. However, in the face of the urgent needs of clinical fundus disease screening, diagnosis and other auxiliary medical care, Existing CAD systems generally can only detect a few major eye diseases. There are few related studies on detecting multiple eye diseases and their performance is average. And most studies directly transfer the features learned by the last convolutional layer of CNN to the classification layer.

A problem: (1) It is impossible to effectively avoid the interference of repeated fundus image background information, and ignores the subtle changes in the fundus image that may represent the diseased tissue; (2) The core operation of CNN is the convolution kernel, which has local sensitivity. However, in the fundus image The lesion area may be discontinuous, so it is difficult for CNN to grasp the global features and extract them. This study contributes a new solution to the two limitations of the above-mentioned fundus image classification research: through a series of algorithm improvements to the convolutional neural network (CNN), the CNN-Trans model is proposed, which is an attention mechanism and feature fusion. The detailed architecture and principles of the model are explained in Section 3.2 of this article., considering the problem that the existing CAD system detects a small number of fundus diseases, Our model also conducted related 7 classification experiments on the fundus image data set: age-related macular degeneration (Age\_degeneration), cataract (Cataract), diabetic retinopathy (Diabetes), glaucoma (Glaucoma), hypertension complications (Hypertension), pathological myopia (Myopia) and normal (Normal), the classification accuracy reached 80.68 %, which achieved better performance than other similar studies.

**CHAPTER 2**

**RELATED WORK**

**Literature Survey on Fundus Image Processing and Ophthalmic Diagnosis**

Fundus image processing plays a crucial role in ophthalmic diagnosis, particularly in detecting and predicting eye diseases such as diabetic retinopathy, cataracts, and myopia. The following literature survey provides insights into the advancements in fundus image processing, machine learning applications, and deep learning techniques for ophthalmic diagnosis based on the given references.

**Fundus Image Registration and Processing Techniques**

Yu et al. [1] discuss the progress in fundus image registration technology, which is essential for aligning images taken at different times or from different perspectives. Their study focuses on different registration methods, including intensity-based and feature-based techniques, to enhance the accuracy of image alignment, thus facilitating disease diagnosis.

The book by CRC Press [2] explores signal processing and machine learning applications in biomedical big data, including fundus imaging. It highlights how advanced computational techniques help in image enhancement, segmentation, and feature extraction, leading to improved diagnostic accuracy.

**Deep Learning in Fundus Image Analysis**

Sengupta et al. [3] provide a comprehensive review of deep learning applications in fundus image processing for ophthalmic diagnosis. Their study covers convolutional neural networks (CNNs) and transfer learning approaches used in detecting various retinal diseases. The authors emphasize the importance of large annotated datasets and robust model architectures in achieving high diagnostic accuracy.

Socia et al. [4] propose machine learning-based detection of trachoma using fundus images. Their study employs convolutional and traditional machine learning algorithms to classify images with high sensitivity and specificity. This research demonstrates the potential of artificial intelligence in automating disease detection in ophthalmology.

**Epidemiology and Disease Detection**

Foster and Jiang [5] discuss the epidemiology of myopia, analyzing various factors contributing to its prevalence. Their research focuses on population-based studies and genetic predispositions that influence the development of myopia.

Faizal et al. [6] present an automated cataract detection system using adaptive thresholding and a fine-tuned InceptionV3 model. Their method significantly improves early cataract diagnosis by leveraging deep learning-based feature extraction and classification techniques applied to anterior segment eye images.

Qureshi et al. [7] review recent developments in diabetic retinopathy detection methods. The paper discusses feature-based approaches and deep learning models for early detection and grading of diabetic retinopathy, highlighting the advantages of automated diagnostic systems over traditional manual assessments.

**Classification of Fundus Images Using Deep Learning**

Chea and Nam [8] propose a deep learning-based classification system for fundus images to detect various eye diseases. Their study evaluates CNN architectures trained on large datasets, demonstrating improved diagnostic accuracy and reliability compared to conventional methods.

Wen et al. [9] explore the use of retinal images for artificial intelligence-driven kidney disease detection. The study highlights how retinal biomarkers can serve as indicators of systemic diseases, opening new avenues for AI-assisted diagnosis beyond ophthalmology.

Choi et al. [10] develop a multi-categorical deep learning neural network for classifying retinal images. Their pilot study shows that deep learning models can effectively differentiate multiple eye conditions, even with small datasets, suggesting the potential for future improvements with larger, more diverse datasets.

Diaz-Pinto et al. [11] extensively validate CNNs for automatic glaucoma assessment using fundus images. Their study highlights the effectiveness of deep learning in diagnosing glaucoma and achieving high accuracy in automated detection.

Balaji et al. [12] compare the foveal avascular zone in diabetic retinopathy, high myopia, and normal fundus images, providing insights into the structural variations among these conditions through ophthalmic imaging analysis.

Junayed et al. [13] introduce CataractNet, an automated cataract detection system using deep learning for fundus images, demonstrating high performance in identifying cataracts with minimal manual intervention.

Butt et al. [14] employ hybrid deep learning features to detect diabetic retinopathy from fundus images, demonstrating how feature fusion techniques enhance the precision and recall of automated diagnostic systems.

Shyamalee and Meedeniya [15] develop a CNN-based fundus image classification system for glaucoma identification, showcasing the potential of deep learning in detecting early-stage glaucoma.

Ilesanmi et al. [16] conduct a systematic review of retinal fundus image segmentation and classification methods using CNNs, summarizing recent advancements and challenges in the field.

Lu et al. [17] propose a multilabel classification method for multiple fundus diseases using a CNN with squeeze-and-excitation attention, improving the interpretability of deep learning models in ophthalmic diagnosis.

Wang et al. [18] introduce COVIDX-LwNet, a lightweight network ensemble model for COVID-19 detection using chest X-ray images, illustrating the adaptability of deep learning models across different medical imaging domains.

Tartaglione et al. [19] explore the challenges of detecting COVID-19 from chest X-rays with deep learning, emphasizing the difficulties posed by small datasets and class imbalances.

Chollet [20] presents Xception, a deep learning model utilizing depthwise separable convolutions, which has been widely adopted in medical image analysis, including ophthalmic imaging.

Greff et al. [21] conducted an in-depth analysis of the Long Short-Term Memory (LSTM) architecture, investigating the search space and optimization challenges in training LSTMs for sequential tasks. Their study provided insights into the advantages of LSTMs in capturing long-range dependencies, leading to improvements in various neural network applications.

Hou et al. [22] introduced Coordinate Attention, an efficient attention mechanism designed to enhance feature representation in mobile networks. Their work demonstrated how coordinate attention could improve the performance of deep networks, particularly in mobile-friendly architectures, by capturing long-range dependencies while preserving spatial details.

Dosovitskiy et al. [23] proposed the Vision Transformer (ViT), a groundbreaking deep learning model that treats images as sequences of patches, similar to the token-based approach in NLP. Their findings indicated that transformers could outperform conventional CNNs when trained on large-scale datasets, revolutionizing image classification tasks.

Carion et al. [24] presented an end-to-end object detection model using transformers, termed DETR. This work introduced a novel way of formulating object detection as a direct set prediction problem, leading to improved accuracy and robustness against variations in object scale and occlusion.

Joshi and Masilamani [25] proposed a transfer learning-based approach for detecting abnormal fundus images. Their model leveraged pre-trained deep learning architectures to achieve high classification accuracy, making it a valuable tool for early diagnosis of retinal disorders.

Raza et al. [26] employed the Inception-V4 deep learning model for classifying eye diseases and detecting cataracts using digital fundus imaging. Their study emphasized the importance of automated diagnostic tools in ophthalmology, highlighting the potential of deep learning in medical image analysis.

Lai et al. [27] investigated the application of CNNs in cataract detection using digital camera images. Their work showcased the feasibility of leveraging deep learning for automated cataract screening, improving accessibility to early diagnosis and treatment.

Smitha and Jidesh [28] utilized a semi-supervised Generative Adversarial Network (GAN) for classifying multiple retinal disorders from enhanced fundus images. Their approach demonstrated the effectiveness of GANs in improving classification accuracy in medical imaging tasks.

Pan et al. [29] developed a classification model using Inception V3 and ResNet-50 for early diagnostics of fundus diseases. Their research emphasized the potential of combining multiple deep learning architectures to achieve superior classification performance.

Shamsan et al. [30] proposed an automatic classification system for color fundus images to predict various eye disease types based on hybrid feature extraction methods. Their approach demonstrated improved accuracy and robustness in disease classification.

Ali et al. [31] introduced AMDNet23, a hybrid CNN-LSTM deep learning approach with enhanced preprocessing techniques for detecting Age-Related Macular Degeneration (AMD). Their work highlighted the effectiveness of combining CNNs with LSTMs for sequential feature learning in medical image analysis.

Selvaraju et al. [32] proposed Grad-CAM, a gradient-based visualization technique for deep learning models. Their work enabled interpretability in CNN-based models, providing valuable insights into decision-making processes in deep networks.

Carion et al. [33] further refined object detection methodologies using transformers, demonstrating their efficiency in end-to-end detection tasks. Their study paved the way for more advanced transformer-based vision models.

Chen et al. [34] explored generative pretraining from pixels, highlighting the effectiveness of self-supervised learning techniques in training large-scale deep learning models for image analysis tasks.

Dosovitskiy et al. [35] expanded on the capabilities of Vision Transformers, reinforcing their dominance in image classification tasks and emphasizing their scalability and performance improvements over CNNs.

Parikh et al. [36] introduced a decomposable attention model for natural language inference, demonstrating its effectiveness in text classification and interpretation tasks. Their approach contributed significantly to the development of attention mechanisms in deep learning.

Cheng et al. [37] examined LSTM-based networks for machine reading, emphasizing their ability to understand complex textual structures and dependencies. Their work showcased the adaptability of LSTMs in various NLP applications.

Gal and Ghahramani [38] proposed using dropout as a Bayesian approximation to represent model uncertainty in deep learning. Their study contributed to the understanding of uncertainty quantification in neural networks, improving robustness and generalization capabilities.

The reviewed literature highlights significant advancements in fundus image processing, with a strong focus on deep learning and machine learning applications. The studies demonstrate the increasing role of AI-driven techniques in enhancing diagnostic accuracy and automating disease detection in ophthalmology. Future research should aim to improve model generalization, address dataset limitations, and integrate multimodal medical data for comprehensive disease assessment.

**CHAPTER 3**

**INTRODUCTION TO MACHINE LEARNING AND DEEP LEARNING**

**3.1 Introduction**

Over the past decade, artificial intelligence (AI) has become a popular subject both within and outside of the scientific community; an abundance of articles in technology and non-technology-based journals have covered the topics of machine learning (ML), deep learning (DL), and AI.1–6 Yet there still remains confusion around AI, ML, and DL. The terms are highly associated, but are not interchangeable. In this review, we (attempt to) forgo technical jargon to better explain these concepts to a clinical audience.

In 1956, a group of computer scientists proposed that computers could be programmed to think and reason, “that every aspect of learning or any other feature of intelligence [could], in principle, be so precisely described that a machine [could] be made to simulate it.”7 They described this principle as “artificial intelligence.”7 Simply put, AI is a field focused on automating intellectual tasks normally performed by humans, and ML and DL are specific methods of achieving this goal. That is, they are within the realm of AI (Fig. 1). However, AI includes approaches that do not involve any form of “learning.” For instance, the subfield known as symbolic AI focuses on hardcoding (i.e., explicitly writing) rules for every possible scenario in a particular domain of interest. These rules, written by humans, come from a priori knowledge of the particular subject and task to be completed. For example, if one were to program an algorithm to modulate room temperature of an office, he or she likely already know what temperatures are comfortable for humans to work in and would program the room to cool if temperatures rise above a specific threshold and heat if they drop below a lower threshold. Although symbolic AI is proficient at solving clearly defined logical problems, it often fails for tasks that require higher-level pattern recognition, such as speech recognition or image classification. These more complicated tasks are where ML and DL methods perform well. This review summarizes machine learning and deep learning methodology for the audience without an extensive technical computer programming background.

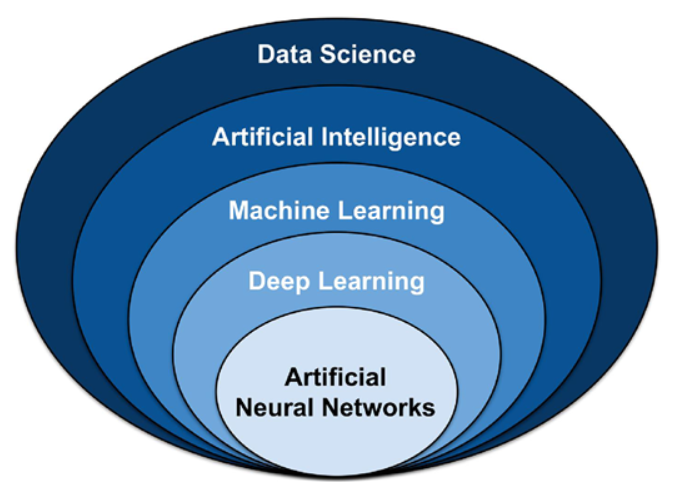


Fig 3.1 Umbrella of select data science techniques.

Artificial intelligence (AI) falls within the realm of data science, and includes classical programming and machine learning (ML). ML contains many models and methods, including deep learning (DL) and artificial neural networks (ANN).

**Methods**

We conducted a literature search in PubMed for articles that were pertinent to leading artificial intelligence methods being utilized in medical research. Selection of articles was at the sole discretion of the authors. The goal of our literature search was to provide the nontechnical readers a layman's explanation of the machine learning methods being used in medicine today.

We found many articles that were pertinent to the main AI methods being used in medicine today.

**Introduction to Machine Learning**

ML is a field that focuses on the learning aspect of AI by developing algorithms that best represent a set of data. In contrast to classical programming (Fig. 2A), in which an algorithm can be explicitly coded using known features, ML uses subsets of data to generate an algorithm that may use novel or different combinations of features and weights than can be derived from first principles (Fig. 2B).8,9 In ML, there are four commonly used learning methods, each useful for solving different tasks: supervised, unsupervised, semisupervised, and reinforcement learning.8–10 To better understand these methods, they will be defined via an example of a hypothetical real estate company that specializes in predicting housing prices and features associated with those houses.

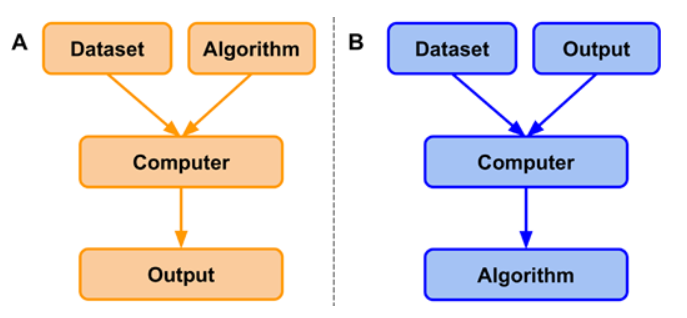


Figure 2. Classical programming versus machine learning paradigm.

(A) In classical programming, a computer is supplied with a dataset and an algorithm. The algorithm informs the computer how to operate upon the dataset to create outputs.

(B) In machine learning, a computer is supplied with a dataset and associated outputs. The computer learns and generates an algorithm that describes the relationship between the two. This algorithm can be used for inference on future datasets.

**Supervised Learning**

Suppose the real estate company would like to predict the price of a house based on specific features of the house. To begin, the company would first gather a dataset that contains many instances.8,9,11 Each instance represents a singular observation of a house and associated features. Features are the recorded properties of a house that might be useful for predicting prices (e.g., total square-footage, number of floors, the presence of a yard).8,9,11 The target is the feature to be predicted, in this case the housing price.8,9,11 Datasets are generally split into training, validation, and testing datasets (models will always perform optimally on the data they are trained on).8,9 Supervised learning uses patterns in the training dataset to map features to the target so that an algorithm can make housing price predictions on future datasets. This approach is supervised because the model infers an algorithm from feature-target pairs and is informed, by the target, whether it has predicted correctly.8–10 That is, features, x, are mapped to the target, Y, by learning the mapping function, f, so that future housing prices may be approximated using the algorithm Y  =  f(x). The performance of the algorithm is evaluated on the test dataset, data that the algorithm has never seen before.8,9 The basic steps of supervised machine learning are (1) acquire a dataset and split it into separate training, validation, and test datasets; (2) use the training and validation datasets to inform a model of the relationship between features and target; and (3) evaluate the model via the test dataset to determine how well it predicts housing prices for unseen instances. In each iteration, the performance of the algorithm on the training data is compared with the performance on the validation dataset. In this way, the algorithm is tuned by the validation set. Insofar as the validation set may differ from the test set, the performance of the algorithm may or may not generalize. This concept will be discussed further in the section on performance evaluation.

The most common supervised learning tasks are regression and classification.8–10 Regression involves predicting numeric data, such as test scores, laboratory values, or prices of an item, much like the housing price example.8–10 Classification, on the other hand, entails predicting to which category an example belongs.8–10 Sticking with the previous example, imagine that rather than predicting exact housing prices in a fluctuating market, the real estate company would now like to predict a range of prices for which a house will likely sell, such as (0, 125K), (125K, 250K), (250K, 375K), and (375K, ∞). To accomplish this, data scientists would transform the numeric target variable into a categorical variable by binning housing prices into separate classes. These classes would be ordinal, meaning that there is a natural order associated with the categories.9 However, if their task was to determine whether houses had wood, plastic, or metal siding, classes would be nominal; they are independent of one another and have no natural order.9

**Unsupervised Learning**

In contrast to supervised learning, unsupervised learning aims to detect patterns in a dataset and categorize individual instances in the dataset to said categories.8–10 These algorithms are unsupervised because the patterns that may or may not exist in a dataset are not informed by a target and are left to be determined by the algorithm. Some of the most common unsupervised learning tasks are clustering, association, and anomaly detection.8–10 Clustering, as the name suggests, groups instances in a dataset into separate clusters based upon specific combinations of their features.8–10 Say the real estate company now uses a clustering algorithm on its dataset and it finds three distinct clusters. Upon further investigation, it might find that the clusters represent the three separate architects responsible for designing the homes in their dataset, which is a feature that was not present in the training dataset.

**Semisupervised Learning**

Semisupervised learning can be thought of as the “happy medium” between supervised and unsupervised learning and is particularly useful for datasets that contain both labeled and unlabeled data (i.e., all features are present, but not all features have associated targets).10 This situation typically arises when labeling images become time-intensive or cost-prohibitive. Semisupervised learning is often used for medical images, where a physician might label a small subset of images and use them to train a model. This model is then used to classify the rest of the unlabeled images in the dataset. The resultant labeled dataset is then used to train a working model that should, in theory, outperform unsupervised models.10

**Reinforcement Learning**

Finally, reinforcement learning is the technique of training an algorithm for a specific task where no single answer is correct, but an overall outcome is desired.9,10 It is arguably the closest attempt at modeling the human learning experience because it also learns from trial and error rather than data alone.9,10 Although reinforcement learning is a powerful technique, its applications in medicine are currently limited and thus will be presented with a new example. Imagine one would like to train an algorithm to play the video game Super Mario Bros, where the purpose of the game is to move the character Mario from the left side of the screen to the right side in order to reach the flag pole at the end of each level while avoiding hazards such as enemies and pits. There is no correct sequence of controller inputs; there are sequences that lead to a win and those that do not. In reinforcement learning, an algorithm would be allowed to “play” on its own. It would attempt many different controller inputs and when it finally moves Mario forward (without receiving damage), the algorithm is “rewarded” (i.e., the behavior is reinforced). Through this process, the algorithm begins to learn what behavior is desired (e.g., moving forward is better than moving backward, jumping over enemies is better than running into them). Eventually, the algorithm learns how to move from start to finish. Although reinforcement has its place in the field of computer science and machine learning, it has yet to make a substantial impact in clinical medicine.

Performance Evaluation

To maximize the chance of generalizability to the performance of the algorithm on unseen data, the training dataset is usually split into a slightly smaller training dataset and a separate validation dataset.8,9 Metrics used for evaluation of a model depend upon the model itself and whether it is in the training or testing phase. The validation dataset is meant to mimic the test dataset and helps data scientists tune an algorithm by identifying when a model may generalize well and work in a new population. Because the validation dataset is a small sample of the true (larger) population, it may not accurately represent the population itself due to an unknown sampling bias. Therefore, model performance and generalizability should not be assessed via validation set performance. It is conceivable that a data scientist could create a validation dataset with an unknown bias and use it to tune a model. Although the model might perform well on the validation dataset, it would likely not perform well on the much larger test dataset (i.e., it would not be a generalizable model)

Typically, model performance is monitored via some form of accuracy on the training and validation datasets during this phase. So long as the accuracy of the model on the training set (X%) and validation set (Y%) are increasing and converging after each training iteration, the model is considered to be learning. If both converge, but do not increase (e.g., X converges on Y at 50%), the model is not learning and may be underfit to the data, that is, it may not have learned enough of the relationship between features and targets in a way that it would be expected to work in another population. Finally, if training performance increases far more than validation set performance (e.g., the model has an accuracy of 99% on the data it was trained on, but only 80% on the validation data), the model is overfit. That is, it has learned features specific to the training dataset population at the expense of generalizability to another population. Although the validation dataset is not specifically used to train the algorithm, it is used to iteratively tune the algorithm. Therefore, the validation dataset is not necessarily a reliable indicator of model performance on unseen data.8,9

Upon completion of the training phase, a data scientist has, ideally, trained a highly generalizable model; however, this must be confirmed via a separate test dataset. In the case of supervised learning, which will be the focus of this review from here on, the performance of a learned model can be evaluated in a number of ways, but is most commonly evaluated based on prediction accuracy (classification) or error and residuals (regression).8,9 As previously mentioned, the test dataset contains instances of the original dataset that have not been seen by the algorithm during the training phase. If the predictive power of a model is strong on the training dataset, but poor on the test dataset, then the model is too specific to the patterns from the training data and is considered to be overfit to the training dataset.8,9 That is, it has memorized patterns rather than learned a generalizable model. An underfit model, on the other hand, is one that performs poorly on both training and test datasets and has neither learned nor memorized the training dataset and still is not generalizable.8,9 An ideally fitted model is one that performs strongly on both datasets, suggesting it is generalizable (i.e., it will perform well on other similar datasets).8,9

With regression models, the average mean squared error (MSE) can be an indicator of model performance.8,9 MSE measures how close a predicted value is to the intended target value. MSE is calculated by summing the differences between predicted values and target values, squaring the results, and dividing by the total number of instances . There are many other measures of performance for regression models that are out of the scope of this review.

For binary classification, the output of the model is a class. However, before the class designation, the probability of an instance belonging to class A or class B is determined.8,9 Normally, this probability threshold is set at 0.5. A receiver operating characteristic curve evaluates a model's true positive rate (TPR; i.e., sensitivity, recall), the number of samples correctly identified as positive divided by the total number of positive samples, versus its false-positive rate (FPR; i.e., 1 - specificity), the number of samples incorrectly identified as positive divided by the total number of negative samples (Fig. 3, Fig. 4A).8,9 Similarly, the precision-recall curve evaluates a model's positive predictive value (PPV; i.e., precision), the number of samples correctly identified as positive divided by the total number of samples identified as positive, versus its recall (Fig. 3.3, Fig. 4B).8,9 Each curve is evaluated across the range of model probability thresholds from 1 to 0, left to right. A receiver operating characteristic curve starts at the point (FPR = 0, TPR = 0), which corresponds to a decision threshold of 1 (every sample is classified as negative, and thus there are no false or true positives). It ends at the point (FPR = 1, TPR = 1), which corresponds to a decision threshold of 0 (where every sample is classified as positive, and thus all points are either truly or falsely labeled positive). The points in between, which create the curve, are obtained by calculating the TPR and FPR for different decision thresholds between 1 and 0, trading off sensitivity (minimizing false negatives) with specificity (minimizing false positives). The area under the curve (AUC) of the receiver operating characteristics curve (AUROC) can be calculated and used as a metric for evaluating the overall performance of a classifier, assuming the classes of the dataset are balanced. If classes are not balanced, the area under the precision-recall curve (AUPR) may be a better metric of model performance because the threshold (set at 0.5 in Fig. 4B) may be adjusted. For example, if a dataset comprised 75% of class A and 25% of class B, the ratio between the two would be computed as the threshold (0.75). In practice, an AUROC value of 0.50 indicates a model that performs no better than chance, and an AUC of 1.00 indicates that the model performs perfectly; the higher the value of the AUC, the stronger the performance of the ML model.8,9 Similarly, an AUPR value at the preset threshold indicates a model that performs no better than chance, and an AUPR value of 1.00 indicates a perfect model.



Figure 3.3 Sensitivity, specificity, positive predictive value, and negative predictive value.

A population (dataset) is represented as circles colored blue if positive or orange if negative. The dataset is input to an algorithm that predicts each instance's class association. If an instance is correctly predicted as positive or negative, it is a true positive (TP) or true negative (TN), respectively. If an instance is incorrectly labeled positive or negative, it is a false positive (FP) or false negative (FN), respectively.

(A) A model with perfect sensitivity and specificity .

(B) A model with perfect sensitivity (ability to correctly classify all positive cases), but poor specificity (ability to correctly classify all negative cases) and (C) a model with perfect specificity, but poor sensitivity. Although a model might have perfect sensitivity (B), it can have many false positives. Similarly, a model with perfect specificity (C) might have many false negatives. Therefore, it is also useful to evaluate the positive predictive value  and the negative predictive value PPV and NPV are also thus dependent on the prevalence of disease in a population.

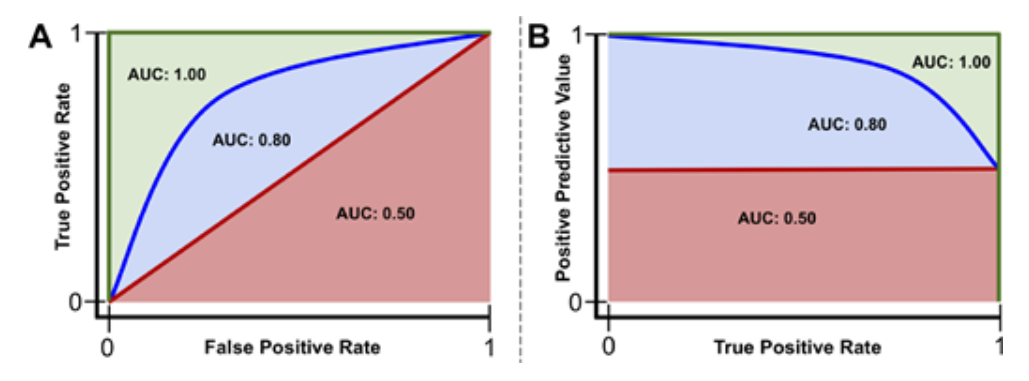


Figure 3.4. Example receiver operating characteristics and precision-recall curves.

Red line: a model that performs no better than chance has an area under the curve (AUC) of the receiver operating characteristics curve (AUROC) of 0.50 or area under the precision-recall curve (AUPR) at the class ratio (red shaded area). Blue line: a model that performs better than chance, but not perfectly, will have an AUC between 0.50 and 1.00 (blue + red shaded areas). Green line: a model that performs perfectly has an AUC of 1.00 (red + blue + green shaded areas).

Classic Machine Learning Methods

There are many machine learning algorithms used in medicine. Described next are some of the most popular to date.

**Linear Regression**

Linear regression is arguably the simplest ML algorithm. The main idea behind regression analysis is to specify a relationship between one or more numeric features and a single numeric target.8,9 Linear regression is an analysis technique used to solve a regression problem by using a straight line to describe a dataset. Univariate linear regression, a regression problem where only a single feature is used for predicting a target value, can be represented in a slope-intercept form: y  =  ax  +  b.8,9 Here, a is a weight describing the slope, which describes how much a line increases on the y-axis for each increase in x. The intercept, b, describes the point where the line intercepts the y-axis. Linear regression models a dataset using this slope-intercept form, where the machine's task is to identify values of a and b such that the determined line is best able to relate the supplied values of x values to the values of y. Multivariate linear regression is similar; however, there are multiple weights in the algorithm, each describing to what degree each feature influences the target

In practice, there is rarely a single function that fits a dataset perfectly. To measure the error associated with a fit, the residuals are measured. Conceptually, residuals are the vertical distances between predicted values, y^, and actual values, y. In machine learning, the cost function is a calculus derived term that aims to minimize errors associated with a model.8,9 The process of minimizing the cost function involves an iterative optimization algorithm known as gradient descent, of which the mathematical calculations involved are outside the scope of this article.8,9,12 In linear regression, the cost function is the previously described MSE. Minimizing this function often obtains estimates of a and b that best model a dataset. All model-based learning algorithms have a cost function, and the goal is to minimize this function to find the best-fit model.8,9

**Logistic Regression**

Logistic regression is a classification algorithm where the goal is to find a relationship between features and the probability of a particular outcome. Rather than using the straight line produced by linear regression to estimate class probability, logistic regression uses a sigmoidal curve to estimate class probability (Fig. 5). This curve is determined by the sigmoid function, . which produces an S-shaped curve that converts discrete or continuous numeric features (x) into a single numerical value (y) between 0 and 1.8,9 The major advantage of this method is that probabilities are bounded between 0 and 1 (i.e., probabilities cannot be negative or greater than 1). It can be either binomial, where there are only two possible outcomes, or multinomial, where there can be three or more possible outcomes.

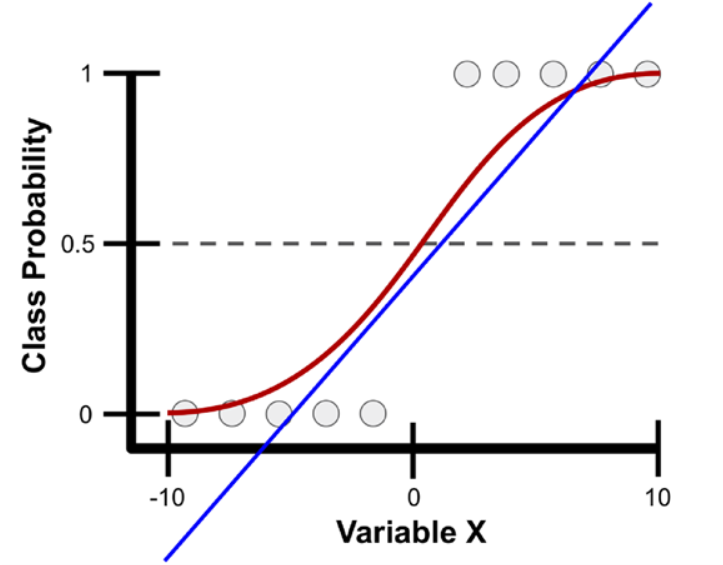


Figure 3.5. Example class probability prediction using linear and logistic regression.

Presented are linear (blue line) and logistic (red line) regression models for predicting the probability of various samples (gray circles) as belonging to a particular class using a single variable, variable X, which ranges from -10 to 10. With logistic regression, variable X is transformed into class probabilities that are bounded between 0 and 1 using the sigmoid function. Simple linear regression attempts to estimate class probabilities, but is not bounded between 0 and 1; thus, it breaks a fundamental law of probability that does not allow for negative probabilities or those greater than 1.

**Decision Trees and Random Forests**

A decision tree is a supervised learning technique, primarily used for classification tasks, but can also be used for regression. A decision tree begins with a root node, the first decision point for splitting the dataset, and contains a single feature that best splits the data into their respective classes (Fig. 3.6). Each split has an edge that connects either to a new decision node that contains another feature to further split the data into homogenous groups or to a terminal node that predicts the class. This process of separating data into two binary partitions is known as recursive partitioning. A random forest is an extension of this method, known as an ensemble method, that produces multiple decision trees.8,9 Rather than using every feature to create every decision tree in a random forest, a subsample of features are used to create each decision tree. Trees then predict a class outcome, and the majority vote among trees is used as the model's final class prediction.

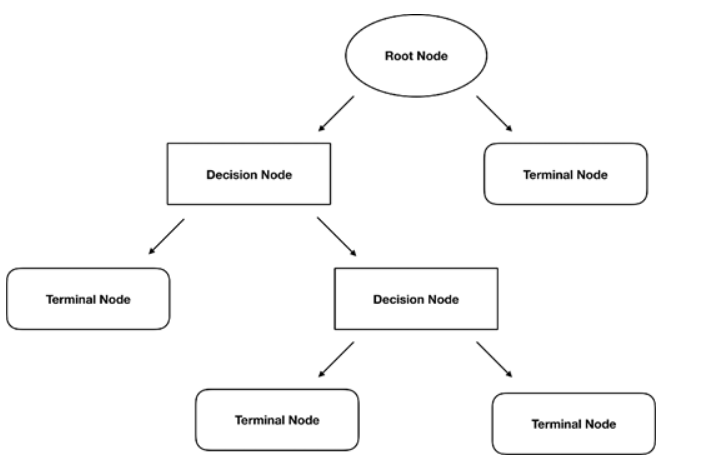


Figure 3.6. Structure of a decision tree.

Splitting of the dataset begins at the root node. Each split connects to either another decision node, which results in further splitting of the data, or a terminal node that predicts the class of the data.

**Classic Machine Learning in Ophthalmology**

Although DL has become a highly popular technique in ophthalmology, there are a multitude of examples of classic ML algorithms being used in the field. Simple linear models have been used to predict patients who would develop advanced age-related macular degeneration and to discern which factors separate patients into who will respond to anti-vascular endothelial growth factor treatment versus those who will not. Random forest algorithms have been used to discover features that are most predictive of progression to geographic atrophy in age-related macular degeneration and find prognostic features for visual acuity outcomes of intravitreal anti-vascular endothelial growth factor treatment. Random forest classifiers have also been applied to diagnose and grade cataracts from ultrasound images, as well as identify patients with glaucoma based on retinal nerve fiber layer and visual field data.

**Neural Networks and Deep Learning**

An artificial neural network (ANN) is a machine learning algorithm inspired by biological neural networks.8,9,21 Each ANN contains nodes (analogous to cell bodies) that communicate with other nodes via connections (analogous to axons and dendrites). Much in the way synapses between neurons are strengthened when their neurons have correlated outputs in a biological neural network (the Hebbian theory postulates that “nerves that fire together, wire together”), connections between nodes in an ANN are weighted based upon their ability to provide a desired outcome.8,9,21

**Feedforward Neural Networks**

A perceptron is a machine learning algorithm that takes in a series of features and their targets as input and attempts to find a line, plane, or hyperplane that separates the classes in a two-, three-, or hyper-dimensional space, respectively.These features are transformed using the sigmoid function (Fig. 3.7A). Thus, this method is similar to logistic regression; however, it only provides class associations, and not the probability of an instance belonging to a class.

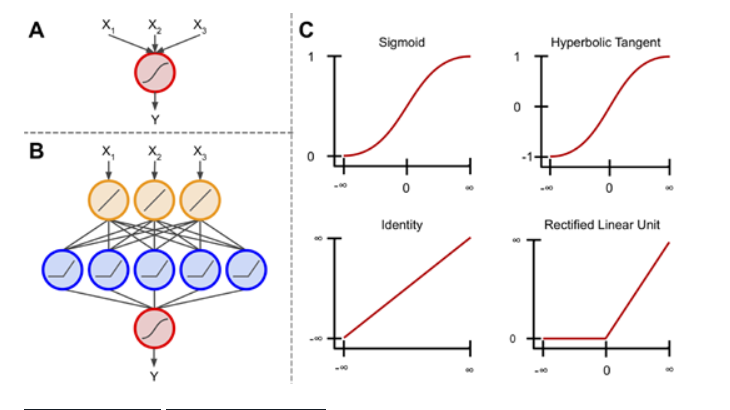


Figure 3.7. Components of a neural network.

(A) The basis of an artificial neural network, the perceptron. This algorithm uses the sigmoid function to scale and transform multiple inputs into a single output ranging from 0 to 1.

(B) An artificial neural network connects multiple perceptron units, so that the output of one unit is used as input to another. Additionally, these units are not limited to using the sigmoid activation function.

(C) Examples of four different activation functions: sigmoid, hyperbolic tangent, identity, and rectified linear unit. The sigmoid scales inputs between 0 and 1 using an S-shaped curved. Similarly, the hyperbolic tangent function uses an S-shaped curve, but scales inputs between -1 and 1. The identity function can multiply its input by any number to produce a linear output. The rectified linear unit is similar to the identity function, however all inputs < 0 are given an output value of 0. There are other activation functions outside of these, but these are arguably.

When multiple perceptrons are connected, the model is referred to as a multilayer perceptron algorithm or an ANN. Commonly, ANNs contain a layer of input nodes, a layer of output nodes, and a number of “hidden layers” between the two.9 In simple ANNs, there exists an input layer between zero and three hidden layers and an output layer, whereas deep neural networks contain tens or even hundreds of hidden layers.9,24 For most tasks, ANNs feed information forward. This is known as a feedforward neural network, meaning information from each node in the previous layer is passed to each node in the next layer, transformed, and passed forward to each node in the next layer (Fig. 7B).9 In recurrent neural networks, which are out of the scope of this paper, information can be passed between nodes within a layer or to previous layers, where their output is operated on and fed forward once again.22

Each layer in an ANN can contain any number of nodes; however, the number of nodes in the output layer typically corresponds to the number of classes being predicted if the goal is multiclass classification, a single node with a sigmoidal activation for binary classification, or a linear activation function if the goal is regression.9,24 These activation functions simply transform a node's input into a desired output (Fig. 7C). Each node in an ANN contains an activation function (not just the output layer; Fig. 7B). These activation functions, although not always linear, do not have to be complex. For instance, the rectified linear unit applies a linear transformation to inputs ≥ 0, and sets inputs < 0 to 0.25 It follows that as inputs proceed through an ANN, they are progressively modified at each layer so that at the final layer they no longer resemble their original state. However, this final representation of the input is, in theory, more predictive of the specified outcome.

**Convolutional Neural Networks**

For image recognition tasks, each input into a feed forward ANN corresponds to a pixel in the image. However, this is not ideal because there are no connections between nodes in a layer. In practice, this means that the spatial context of features in the image are lost.24,26,27 In other words, pixels that are close to one another in an image are likely more correlated than pixels on opposite sides of the image, but a feed forward ANN does not take this into account.

A convolutional neural network (CNN) is a special case of the ANN that overcomes this issue by preserving the spatial relationship between pixels in an image.24,26,27 Rather than using single pixels as input, a CNN feeds patches of an image to specific nodes in the next layer of nodes (rather than all nodes), thereby preserving the spatial context from which a feature was extracted.9,24,26,27 These patches of nodes learn to extract specific features and are known as convolutional filters.

Convolutions are widely used in the realm of image processing, and are often used to blur or sharpen images, or for other tasks such as edge detection.28 A visible-light digital image is simply a single matrix if the image is gray scale or three stacked matrices if the image is color (red, green, and blue color channels). These matrices contain values, typically between 0 and 255, that represent pixels in the image and the intensity of each color channel at each pixel.28 A convolutional filter is a much smaller matrix that is typically square and range in size from 2 × 2 to 9 × 9.28 This filter is passed over the original image and, at each position, element-wise matrix multiplication is performed (Fig. 3.8). The output of this convolution is mapped to a new matrix (a feature map) that contains values corresponding to whether or not the convolutional filter detected a feature of interest.

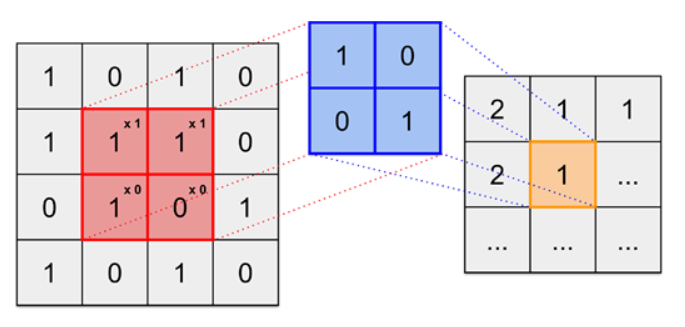


Figure 3.8. Example of a digital image convolved with a filter.

The image (left) is transformed into the feature map (right) via a convolutional filter (center). The convolutional filter is designed to locate diagonal lines running from top left to bottom right of the image. The filter passes over the image in a specified manner and each element in the image (red) is multiplied by the corresponding element in the convolutional filter (blue). The summation of these elements (orange) is output into a new matrix that reports the presence of a diagonal line. The feature map indicates 2 when the specified diagonal line is found, 1 if a portion of it is found, and 0 if none of it is found.

In CNNs, filters are trained to extract specific features from images (e.g., vertical lines, U-shaped objects,) and mark their location on the feature map.A deep CNN then uses the feature map as input for the next layer, which uses new filters to create another new feature map.24,26,27 This can continue for many layers and, as it continues, the extracted features become abstract, but highly useful for prediction. The final features maps are then compressed from their square representations and input to a feedforward ANN, where classification of the image based on the extracted features and textures can occur.This process is referred to as DL.

Aside from image classification tasks, DL has shown promise for image segmentation tasks.1,30,31 Rather than classifying images as a whole, this method aims to identify objects within an image. To accomplish this task, DL classifies individual pixels given surrounding pixel information. For example, in diabetic retinopathy, a segmentation algorithm might segment (outline) the retinal vasculature by assigning probabilities to individual pixels as belonging to a retinal blood vessel or not belonging to a retinal blood vessel. A similar method for breast cancer detection could mark pixels as belonging to a mass or not belonging to a mass, and the output image could be provided to a radiologist for further review.

**Deep Learning in Ophthalmology**

The popularity for DL has especially risen in the field of ophthalmology for image-based diagnostic systems. On the simpler end of visual interpretation tasks, Coyner et al. devised a DL system for automated assessment of retinal fundus image quality with an output of “acceptable” or “not acceptable” based on multiple graded expert labels.3 Presumably, the network learned that the retinal vasculature must be easily distinguishable for an image to be deemed acceptable. In a more complex task, Gulshan et al. demonstrated that DL could classify diabetic retinopathy, in agreement with the Early Treatment for Diabetic Retinopathy Study scale, using only retinal fundus images as input and the consensus diagnoses of multiple clinicians as the “class labels.”2 The presence of features such as microaneurysms, intraretinal hemorrhages, or neovascularization were not supplied to the DL method as signs of diabetic retinopathy. Rather, the DL model either learned these features or learned novel features that aid in the diagnosis of diabetic retinopathy. Further, Brown et al. trained a similar DL network for the diagnosis of plus disease in retinopathy of prematurity. First, an algorithm was trained to segment retinal vasculature into binary vessel maps. Then another DL algorithm was trained to examine the vessel maps and conclude whether the vasculature appeared normal or abnormal.1 This network, too, performs on par or better than most experts in the field. One of the most impressive examples of DL in ophthalmology was conducted by De Fauw et al. Using three-dimensional optical coherence tomography images, a DL framework was trained to not only detect a single disease, but more than 50 common retinal diseases.

**Challenges with DL Models**

In recent years, DL has become a hot topic within the field of medicine given the digital availability of information; however, many challenges still exist. DL is limited by the quantity and quality of data used to train the model. It is difficult to estimate how much data are necessary to sufficiently and reliably train DL systems because it depends both on the quality of the input training data as well as the complexity of the task. Typically, thousands of training examples are required to create a model that is both accurate and generalizable. Thus, developing models for identification of rare diseases, where large datasets may not be readily available, is especially challenging. On the other hand, although one might assume that more data will always lead to better models, if the quality of the training data is imprecise, mislabeled, or somehow systematically different than the test population, training on very large datasets may result in models that do not perform well in real-world scenarios. Furthermore, there is an implicit assumption that datasets are accurately labeled by human graders. Unfortunately, this is often not the case, and noisy and/or missing labels are often a bane for data scientists.

DL methods also suffer from the “black box” problem: input is supplied to the algorithm and an output emerges, but it is not exactly clear what features were identified or how they informed the model output.29,32,33 In contrast, simple linear algorithms, although not always as powerful as DL, are easily interpretable. The computed weights for each feature are supplied upon completion of the training process, which allow for one to interrogate exactly how the model works and possibly discover important predictors that may be useful for prevention of a disease. With deep learning, a complex series of matrix multiplication and abstract filters makes interpretability significantly more challenging.29,32,33 Activation maps, or heatmaps, are methods that attempt to address the “black box” issue by highlighting areas of images that highlight regions of an image that “fire together” with the output classification label.29,32,33 Unfortunately, these methods still require human interpretation, as they are often not examined critically (examples are cherry picked for publication, highly subject to confirmation bias, etc.), and thus this remains an active area of research. For instance, if a DL model classifies a fundus image as having proliferative diabetic retinopathy, a heatmap will highlight feature areas on that fundus image that contributed to the decision of being classified as having proliferative diabetic retinopathy. It is up to the physician to interpret whether these DL model identified features are the same features the physician would use to diagnose the disease, and the implications of such findings.

AI methods have shown to be a promising tool in the field of medicine. Recent work has demonstrated that these methods can develop effective diagnostic and predictive tools to identify various diseases. In the future, AI-based programs may become an integral part of patients’ clinic visits with their ability to assist in diagnosis and management of various diseases. Physicians should take an active approach to understand the theories behind AI and its utility in medicine with the goal of providing optimal patient care.

**CHAPTER 4**

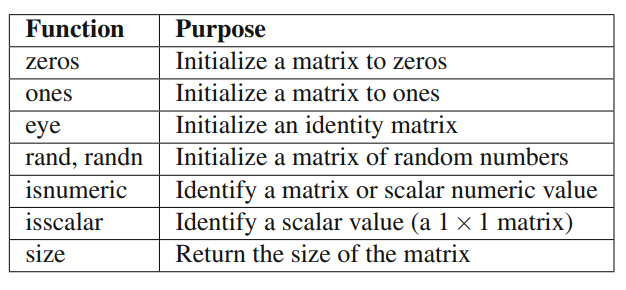
**MATLAB INTRODUCTION**

**Representation of Data for Machine Learning in MATLAB**

**4.1 Introduction to MATLAB Data Types**

4.1.1 Matrices By default, all variables in MATLAB are double-precision matrices. You do not need to declare a type for these variables. Matrices can be multidimensional and are accessed using 1-based indices via parentheses. You can address elements of a matrix using a single index, taken column-wise, or one index per dimension. To create a matrix variable, simply assign a value to it, like this 2×2 matrix a: >> a = [1 2; 3 4]; >> a(1,1) 1 >> a(3) 2 You can simply add, subtract, multiply, and divide matrices with no special syntax. The matrices must be the correct size for the linear algebra operation requested. A transpose is indicated using a single quote suffix, A’, and the matrix power uses the operator ˆ. >> b = a'\*a; >> c = aˆ2; >> d = b + c; By default, every variable is a numerical variable. You can initialize matrices to a given size using the zeros, ones, eye, or rand functions, which produce zeros, ones, identity matrices (ones on the diagonal), and random numbers, respectively. Use isnumeric to identify numeric variables.

Table 4.1 summarizes some key functions for interacting with matrices



**4.1.2 Cell Arrays**

One variable type unique to MATLAB is cell arrays. This is really a list container, and you can store variables of any type in elements of a cell array. Cell arrays can be multidimensional, just like matrices, and are useful in many contexts. Cell arrays are indicated by curly braces, {}. They can be of any dimension and contain any data, including string, structures, and objects. You can initialize them using the cell function, recursively display the contents using celldisp, and access subsets using parentheses just like for a matrix. A short example is below

>> c = cell(3,1);

>> c{1} = 'string';

>> c{2} = false;

>> c{3} = [1 2; 3 4];

>> b = c(1:2);

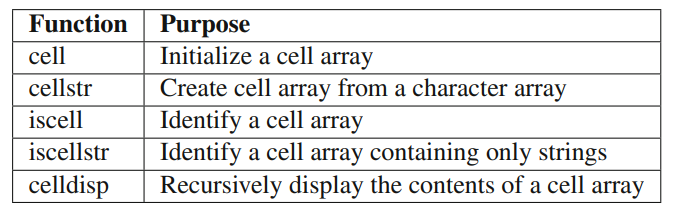
>> celldisp(b)

b{1} = string

b{2} = 0

Using curly braces for access gives you the element data as the underlying type. When you access elements of a cell array using parentheses, the contents are returned as another cell array, rather than the cell contents. MATLAB help has a special section called Comma-Separated Lists that highlights the use of cell arrays as lists. The code analyzer will also suggest more efficient ways to use cell arrays. For instance, Replace a = {b{:} c}; with a = [b {c}]; Cell arrays are especially useful for sets of strings, with many of MATLAB’s string search functions optimized for cell arrays, such as strcmp. Use iscell to identify cell array variables. Use deal to manipulate structure array and cell array contents. Table 4.2 summarizes some key functions for interacting with cell arrays.

Table 4.2: Key Functions for Cell Arrays



**4.1.3 Data Structures**

Data structures in MATLAB are highly flexible, leaving it up to the user to enforce consistency in fields and types. You are not required to initialize a data structure before assigning fields to it, but it is a good idea to do so, especially in scripts, to avoid variable conflicts. Replace d.fieldName = 0; with d = struct; d.fieldName = 0; In fact, we have found it generally a good idea to create a special function to initialize larger structures that are used throughout a set of functions. This is similar to creating a class definition. Generating your data structure from a function, instead of typing out the fields in a script, means you always start with the correct fields. Having an initialization function also allows you to specify the types of variables and provide sample or default data. Remember, since MATLAB does not require you to declare variable types, doing so yourself with default data makes your code that much clearer.

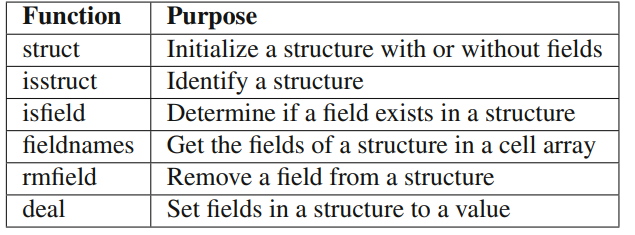
You make a data structure into an array simply by assigning an additional copy. The fields must be in the same order, which is yet another reason to use a function to initialize your structure. You can nest data structures with no limit on depth. d = MyStruct; d(2) = MyStruct; function d = MyStruct d = struct; d.a = 1.0; d.b = 'string'; MATLAB now allows for dynamic field names using variables, that is, structName.(dynamic Expression). This provides improved performance over getfield, where the field name is passed as a string. This allows for all sorts of inventive structure programming. Take our data structure array in the previous code snippet, and let’s get the values of field a using a dynamic field name; the values are returned in a cell array.

>> field = 'a';

>> values = {d.(field)}

values = [1] [1] Use isstruct to identify structure variables and isfield to check for the existence of fields. Note that isempty will return false for a struct initialized with struct, even if it has no fields. Table 4.3 provides key functions for structs.

Table 4.3: Key Functions for Structs



4.1.4 Numerics While MATLAB defaults to doubles for any data entered at the command line or in a script, you can specify a variety of other numeric types, including single, uint8, uint16, uint32, uint64, logical (i.e., an array of booleans). Use of the integer types is especially relevant to using large data sets such as images. Use the minimum data type you need, especially when your data sets are large. 4.1.5 Images MATLAB supports a variety of formats, including GIF, JPG, TIFF, PNG, HDF, FITS, and BMP. You can read in an image directly using imread, which can determine the type automatically from the extension, or fitsread. (FITS stands for Flexible Image Transport System and the interface is provided by the CFITSIO library.) imread has special syntaxes for some image types, such as handling alpha channels for PNG, so you should review the options for your specific images. imformats manages the file format registry and allows you to specify handling of new user-defined types if you can provide read and write functions. You can display an image using either imshow, image, or imagesc, which scales the colormap for the range of data in the image. For example, we use a set of images of cats in Chapter 7, Face Recognition. If we look at the image info for one of these sample images using imfinfo, >> imfinfo('IMG\_4901.JPG')

ans = Filename: 'MATLAB/Cats/IMG\_4901.JPG'

FileModDate: '28-Sep-2016 12:48:15'

FileSize: 1963302 Format: 'jpg'

FormatVersion: '' Width: 3264

Height: 2448

BitDepth: 24

ColorType: 'truecolor' FormatSignature: '’

**NumberOfSamples:** 3 CodingMethod: 'Huffman' CodingProcess: 'Sequential' Comment: {} Make: 'Apple' Model: 'iPhone 6' Orientation: 1 XResolution: 72 YResolution: 72 ResolutionUnit: 'Inch' Software: '9.3.5' DateTime: '2016:09:17 22:05:08' YCbCrPositioning: 'Centered' DigitalCamera: [1x1 struct] GPSInfo: [1x1 struct] ExifThumbnail: [1x1 struct] and we view this image using imshow, it will publish a warning that the image is too big to fit on the screen and that it is displayed at 33%. If we view it using image, there will be a visible set of axes. image is useful for displaying other two-dimensional matrix data as individual elements per pixel. Both functions return a handle to an image object; only the axes’ properties are different. Figure 4.1 shows the resulting figures. Note the labeled axes on the right figure.

>> figure;

hI = image(imread('IMG\_2398\_Zoom.png'))

hI = Image with properties: CData: [680x680x3 uint8]

CDataMapping: 'direct' Show all properties

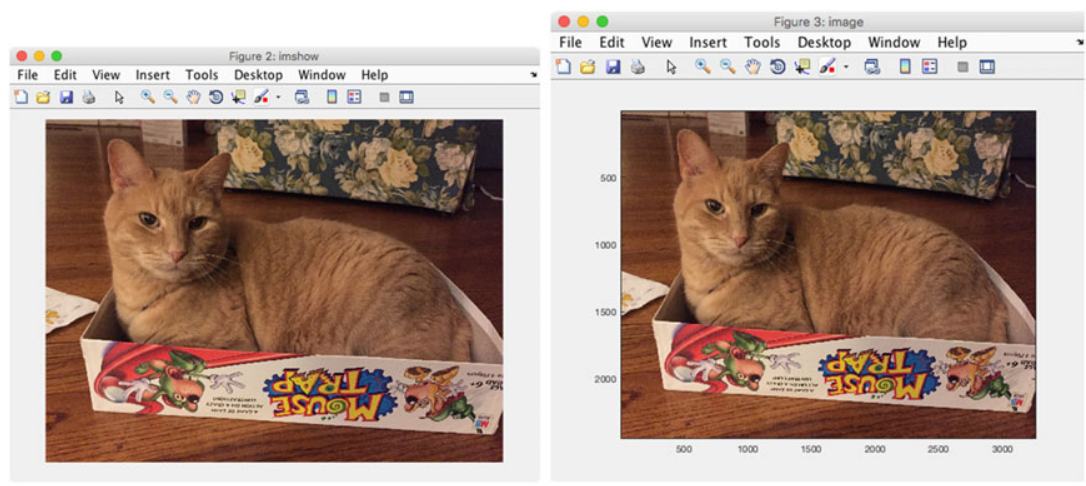


Figure 4.1: Image display options. A figure created using imshow is on the left and a figure using image is on the right

Table 4.4: Key Functions for Image



4.1.6 Datastore Datastores allow you to interact with files containing data that are too large to fit in memory. There are different types of datastores for tabular data, images, spreadsheets, databases, and custom files. Each datastore provides functions to extract smaller amounts of data that do fit in memory for analysis. For example, you can search a collection of images for those with the brightest pixels or maximum saturation values. We will use our directory of cat images as an example.

>> location = pwd

location = /Users/Shared/svn/Manuals/MATLABMachineLearning/MATLAB/Cats

>> ds = datastore(location)

ds = ImageDatastore with properties:

Files: { ' .../Shared/svn/Manuals/MATLABMachineLearning/MATLAB/Cats/ IMG\_0191.png';

' .../Shared/svn/Manuals/MATLABMachineLearning/MATLAB/Cats/ IMG\_1603.png';

' .../Shared/svn/Manuals/MATLABMachineLearning/MATLAB/Cats/ IMG\_1625.png' ... and 19 more }

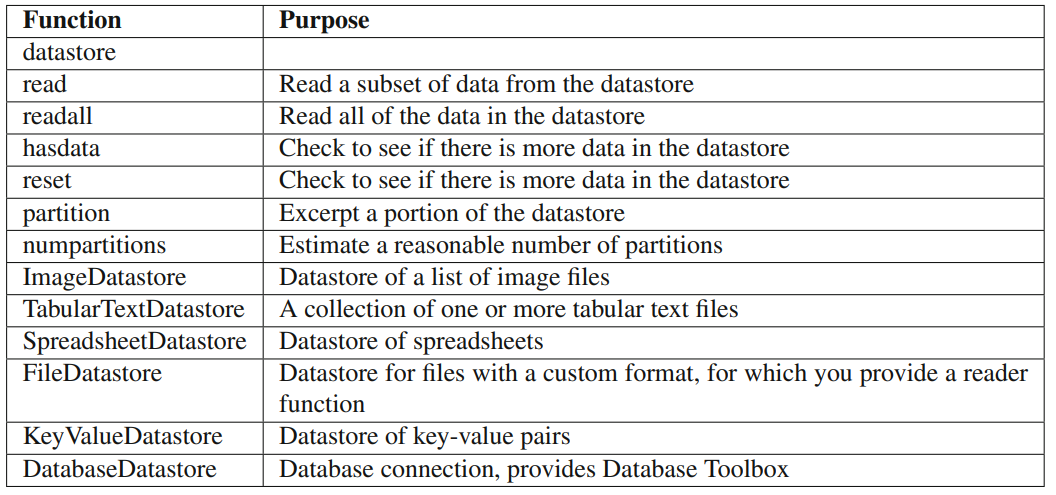
Labels: {} ReadFcn: @readDatastoreImage Once the datastore is created, you use the applicable class functions to interact with it.

Datastores have standard container-style functions like read, partition, and reset. Each type of datastore has different properties.

The DatabaseDatastore requires the Database Toolbox and allows you to use SQL queries.

MATLAB provides the MapReduce framework for working with out-of-memory data in datastores. The input data can be any of the datastore types, and the output is a key-value datastore. The map function processes the datastore input in chunks and the reduce function calculates the output values for each key. mapreduce can be sped up by using it with the MATLAB Parallel Computing Toolbox, Distributed Computer Server, or Compiler. Table 4.5 gives key functions for using datastores.

Table 4.5: Key Functions for Datastore



4.1.7 Tall Arrays Tall arrays are new to release R2016b of MATLAB. They are allowed to have more rows than will fit in memory. You can use them to work with datastores that might have millions of rows. Tall arrays can use almost any MATLAB type as a column variable, including numeric data, cell arrays, strings, datetimes, and categoricals. The MATLAB documentation provides a list of functions that support tall arrays. Results for operations on the array are only evaluated when they are explicitly requested using the gather function. The histogram function can be used with tall arrays and will execute immediately. The MATLAB Statistic and Machine Learning Toolbox™, Database Toolbox, Parallel Computing Toolbox, Distributed Computing Server, and Compiler all provide additional extensions for working with tall arrays. For more information about this new feature, use the following topics in the documentation:

• Tall Arrays

• Analysis of Big Data with Tall Arrays

• Functions That Support Tall Arrays (AZ)

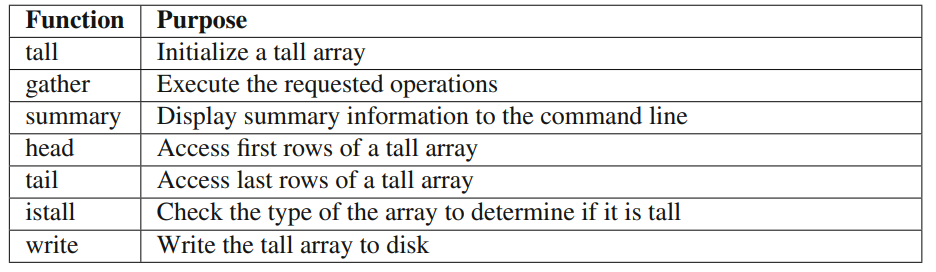
• Index and View Tall Array Elements

• Visualization of Tall Arrays

• Extend Tall Arrays with Other Products

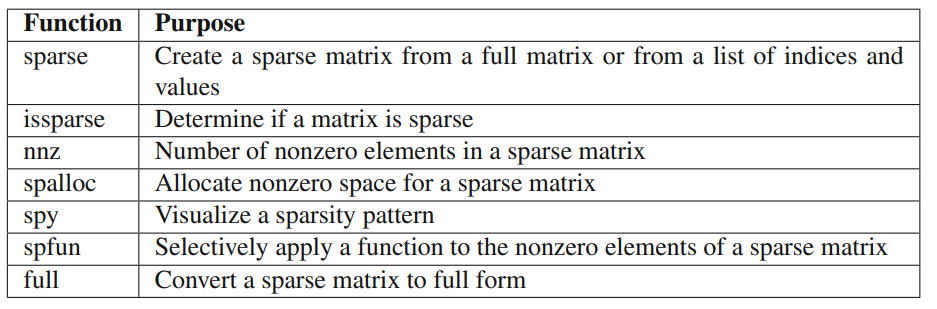
• Tall Array Support, Usage Notes, and Limitations

Table 4.6: Key Functions for Tall Arrays



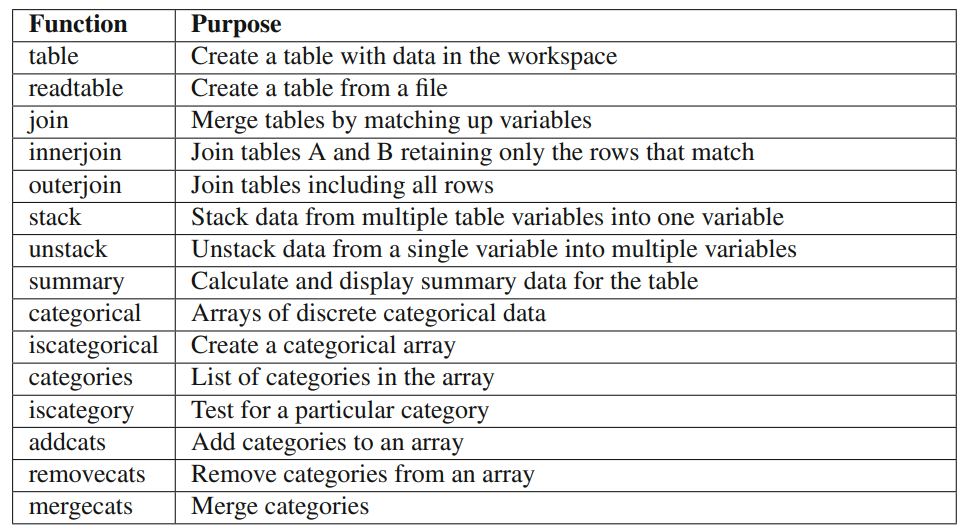
4.1.8 Sparse Matrices Sparse matrices are a special category of matrix in which most of the elements are zero. They appear commonly in large optimization problems and are used by many such packages. The zeros are “squeezed” out and MATLAB stores only the nonzero elements along with index data such that the full matrix can be recreated. Many regular MATLAB functions, such as chol or diag, preserve the sparseness of an input matrix. Table 4.7 gives key functions for sparse matrices.

Table 4.7: Key Functions for Sparse Matrices



4.1.9 Tables and Categoricals Tables were introduced in release R2013 of MATLAB and allow tabular data to be stored with metadata in one workspace variable. It is an effective way to store and interact with data that one might put in, or import from, a spreadsheet. The table columns can be named, assigned units and descriptions, and accessed as one would fields in a data structure, that is, T.DataName. See readtable on creating a table from a file, or try out the Import Data button from the command window. Categorical arrays allow for storage of discrete nonnumeric data, and they are often used within a table to define groups of rows. For example, time data may have the day of the week, or geographic data may be organized by state or county. They can be leveraged to rearrange data in a table using unstack. You can also combine multiple data sets into single tables using join, innerjoin, and outerjoin, which will be familiar to you if you have worked with databases. Table 4.8 lists key functions for using tables

Table 4.8: Key Functions for Tables



**4.1.10 Large MAT-Files**

You can access parts of a large MAT-file without loading the entire file into memory by using the matfile function. This creates an object that is connected to the requested MAT-file without loading it. Data are only loaded when you request a particular variable, or part of a variable. You can also dynamically add new data to the MAT-file. For example, we can load a MAT-file of neural net weights generated in a later chapter.

>> m = matfile('PitchNNWeights','Writable',true)

m = matlab.io.MatFile Properties:

Properties.Source: '/Users/Shared/svn/Manuals/MATLABMachineLearning/ MATLAB/PitchNNWeights.mat' Properties.

Writable: true w: [1x8 double] We can access a portion of the previously unloaded w variable, or add a new variable name, all using this object m.

>> y = m.w(1:4) y = 1111

>> m.name = 'Pitch Weights' m = matlab.io.MatFile

Properties: Properties.Source: '/Users/Shared/svn/Manuals/MATLABMachineLearning/ MATLAB/PitchNNWeights.mat' Properties.Writable: true

name: [1x13 char] w: [1x8 double]

>> d = load('PitchNNWeights')

d = w: [1 1 1 1 1 1 1 1]

name: 'Pitch Weights'

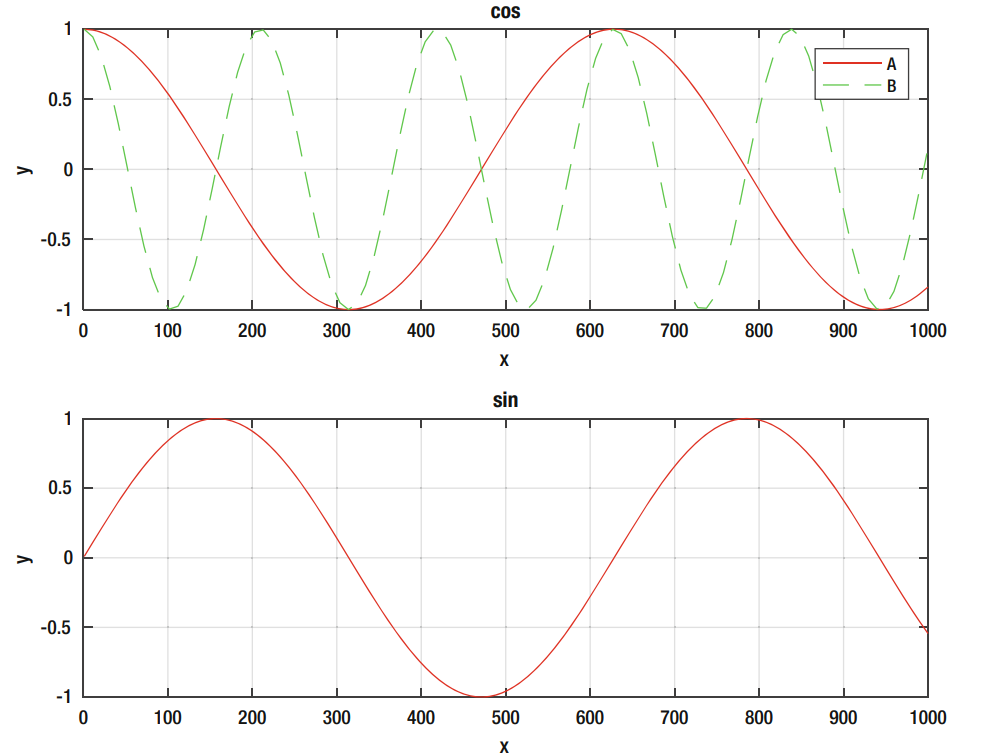
There are some limits to the indexing into unloaded data, such as struct arrays and sparse arrays. Also, matfile requires MAT-files using version 7.3, which is not the default for a generic save operation as of R2016b. You must either create the MAT-file using matfile to take advantage of these features or use the -v7.3’ flag when saving the file.

**4.2 MATLAB Graphics**

Plotting is used extensively in machine learning problems. MATLAB plots can be two or three dimensional. The same data can be represented using many different types of plots. 5.1 Two-Dimensional Line Plots

5.1.1 Problem You want a single function to generate two-dimensional (2D) line graphs, avoiding a long list of code for the generation of each graphic.

5.1.2 Solution Write a single function to take the data and parameter pairs to encapsulate the functionality of MATLAB’s 2D line-plotting functions. An example of a plot created with a single line of code is shown in Figure 5.1

****

**Machine Learning Examples in MATLAB**

**6.1 Introduction**

The remainder of the book provides machine learning examples in MATLAB that span the technologies discussed. Each example provides a useful application in its own right. Full source code is provided. In each case the theory behind the code is provided. References for further study are provided. Each example is self-contained and addresses one of the autonomous learning technologies discussed earlier in the book. You can jump around and try the examples that interest you the most. As we explained earlier, autonomous learning is a huge field. There are many benefits from knowing all aspects of the field. Those with experience in any one of the applications may find the examples to be straightforward. Topics outside your area of expertise will be more challenging. Much like cross-training in the gym, working in other areas will help you in your own area of expertise.

**6.2 Machine Learning**

We present three types of machine learning algorithms. In each case we present a simple algorithm to achieve the desired results.

**6.2.1 Neural Networks**

This example will use a neural network to classify digits. We will start with a set of six digits and create a training set by adding noise to the digital images. We will then see how well our learning network performs at identifying a single digit, and then add more nodes and outputs to identify multiple digits with one network. Classifying digits is one of the oldest uses of machine learning. The U.S. Post Office introduced zip code reading years before machine learning started hitting the front pages of all the newspapers! Earlier digit readers required block letters written in well-defined spots on a form. Reading digits off any envelope is an example of learning in an unstructured environment.

**6.2.2 Face Recognition**

Face recognition is available in almost every photo application. Many social media sites, such as Facebook and Google Plus, also use face recognition. Cameras have built-in face recognition, though not identification, to help with focusing when taking portraits. Our goal is to get the algorithm to match faces, not classify them. Data classification is covered in the next chapter. There are many algorithms for face identification, and commercial software can use multiple algorithms. In this application, we pick a single algorithm and use it to identify one face in a set of photographs—of cats. Face recognition is a subset of general image recognition. The chapter on neural networks, Chapter 9, gives another example. Our example of face recognition works within a structured environment. The pictures are all taken from the front and the picture only shows the head. This makes the problem much easier to solve. 6.2.3 Data Classification This example uses a decision tree to classify data. Classifying data is one of the most widely used areas of machine learning. In this example, we assume that two data points are sufficient to classify a sample and determine to which group it belongs. We have a training set of known data points with membership in one of three groups. We then use a decision tree to classify the data. We’ll introduce a graphical display to make understanding the process easier. With any learning algorithm it is important to know why the algorithm made its decision. Graphics can help you explore large data sets when columns of numbers aren’t terribly helpful.

**6.3 Control Feedback**

control algorithms inherently learn about the environment through measurements used for control. These chapters show how control algorithms can be extended to effectively design themselves using measurements. The measurements may be the same as used for control but the adaptation, or learning, happens more slowly than the control response time. An important aspect of control design is stability. A stable controller will produce bounded outputs for bounded inputs. It will also produce smooth, predictable behavior of the system that is controlled. An unstable controller will typically experience growing oscillations in the quantities (such as speed or position) that are controlled. In these chapters we explore both the performance of learning control and the stability of such controllers.

**6.3.1 Kalman Filters**

The Kalman filters chapter, Chapter 10, shows how Kalman filters allow you to learn about dynamical systems for which we already have a model. This chapter provides an example of a variable-gain Kalman filter for a spring system. That is a system with a mass connected to its base via a spring and a damper. This is a linear system. We write the system in discrete time. This provides an introduction to Kalman filtering. We show how Kalman filters can be derived from Bayesian statistics. This ties it into many machine learning algorithms. Originally, the Kalman filter, developed by R. E. Kalman, C. Bucy, and R. Battin, was not derived in this fashion. The second section adds a nonlinear measurement. A linear measurement is a measurement proportional to the state (in this case position) it measures. Our nonlinear measurement will be the angl a tracking device that points at the mass from a distance from the line of movement. One way is to use an unscented Kalman filter (UKF) for state estimation. The UKF lets us use a nonlinear measurement model easily. The last part of the chapter describes the UKF configured for parameter estimation. This system learns the model, albeit one that has an existing mathematical model. As such, it is an example of modelbased learning. In this example the filter estimates the oscillation frequency of the spring-mass system. It will demonstrate how the system needs to be stimulated to identify the parameters.

**6.3.2 Adaptive Control**

Adaptive control is a branch of control systems in which the gains of the control system change based on measurements of the system. A gain is a number that multiplies a measurement from a sensor to produce a control action such as driving a motor or other actuator. In a nonlearning control system, the gains are computed prior to operation and remain fixed. This works very well most of the time since we can usually pick gains so that the control system is tolerant of parameter changes in the system. Our gain “margins” tell us how tolerant we are to uncertainties in the system. If we are tolerant to big changes in parameters, we say that our system is robust. Adaptive control systems change the gain based on measurements during operation. This can help a control system perform even better. The better we know a system’s model, the tighter we can control the system. This is much like driving a new car. At first you have to be cautious driving a new car because you don’t know how sensitive the steering is to turning the wheel or how fast it accelerates when you depress the gas pedal. As you learn about the car you can maneuver it with more confidence. If you didn’t learn about the car, you would need to drive every car in the same fashion. This chapter starts with a simple example of adding damping to a spring using a control system. Our goal is to get a specific damping time constant. For this we need to know the spring constant. Our learning system uses a fast Fourier transform to measure the spring constant. We’ll compare it to a system that does know the spring constant. This is an example of tuning a control system. The second example is model reference adaptive control of a first-order system. This system automatically adapts so that the system behaves like the desired model. This is a very powerful method and applicable to many situations. The third example is longitudinal control of an aircraft. We can control the pitch angle using the elevators. We have five nonlinear equations for the pitch rotational dynamics, velocity in the x-direction, velocity in the z-direction, and change in altitude. The system adapts to changes in velocity and altitude. Both change the drag and lift forces and the moments on the aircraft and also change the response to the elevators. We use a neural net as the learning element of our control system. This is a practical problem applicable to all types of aircraft ranging from drones to high-performance commercial aircraft. Our last example will be ship steering control. Ships use adaptive control because it is more efficient than conventional control. This example demonstrates how the control system adapts and how it performs better than its nonadaptive equivalent. This is an example of gain scheduling. 6.4 Artificial Intelligence Only one example of artificial intelligence is included in the book. This is really a blending of Bayesian estimation and controls. Machine learning is an offshoot of artificial intelligence so all the machine learning examples could also be considered examples of artificial intelligence.

6.4.1 Autonomous Driving and Target Tracking Autonomous driving is an area of great interest to automobile manufacturers and to the general public. Autonomous cars are driving the streets today but are not yet ready for general use by the public. There are many technologies involved in autonomous driving. These include 1. Machine vision: turning camera data into information useful for the autonomous control system 2. Sensing: using many technologies including vision, radar, and sound to sense the environment around the car 3. Control: using algorithms to make the car go where it is supposed to go as determined by the navigation system 4. Machine learning: using massive data from test cars to create databases of responses to situations 5. GPS navigation: blending GPS measurements with sensing and vision to figure out where to go 6. Communications/ad hoc networks: talking with other cars to help determine where they are and what they are doing All of the areas overlap. Communications and ad hoc networks are used with GPS navigation to determine both absolute location (what street and address correspond to your location) and relative navigation (where you are with respect to other cars). This example explores the problem of a car being passed by multiple cars and needing to compute tracks for each one. We are really addressing just the control and collision avoidance problem. A singlesensor version of track-oriented multiple-hypothesis testing is demonstrated for a single car on a twolane road. The example includes MATLAB graphics that make it easier to understand the thinking of the algorithm. The demo assumes that the optical or radar preprocessing has been done and that each target is measured by a single “blip” in two dimensions. An automobile simulation is included. It involves cars passing the car that is doing the tracking. The passing cars use a passing control system that is in itself a form of machine intelligence. This chapter uses a UKF for the estimation of the state. This is the underlying algorithm that propagates the state (that is, advances the state in time in a simulation) and adds measurements to the state. A Kalman filter, or other estimator, is the core of any target tracking system. The section will also introduce graphics aids to help you understand the tracking decision process. When you implement a learning system, you want to make sure it is working the way you think it should, or understand why it is working the way it does.

**Convolution**

We create an n-x-n mask that we apply to the input matrix. The matrix dimensions are m x m, where m is greater than n. We start in the upper left corner of the matrix. We multiply the mask times the corresponding elements in the input matrix and do a double sum. That is the first element of the convolved output. We then move it column by column until the highest column of the mask is aligned with the highest column of the input matrix. We then return it to the first column and increment the row. We continue until we have traversed the entire input matrix and our mask is aligned with the maximum row and maximum column. The mask represents a feature. In effect, we are seeing if the feature appears in different areas of the image. We can have multiple masks. There are one bias and one weight for each element of the mask for each feature. In this case, instead of 16 sets of weights and biases, we only have 4. For large images, the savings can be substantial. In this case the convolution works on the image itself. Convolutions can also be applied to the output of other convolutional layers or pooling layers.

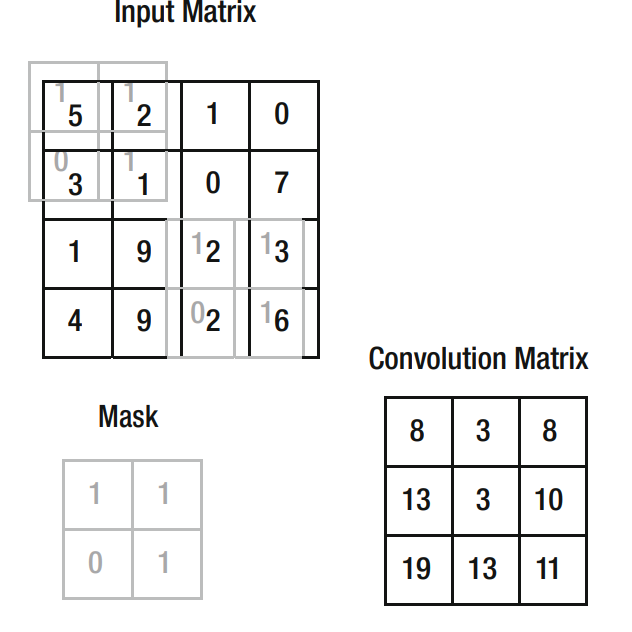
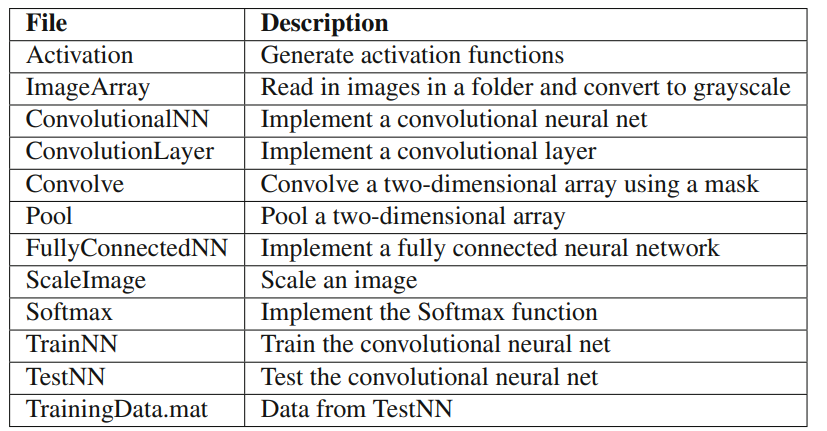


Fig Convolution process showing the mask at the beginning and end of the process.



**Software description**

MATLAB? MATLAB® is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include • Math and computation • Algorithm development • Data acquisition • Modeling, simulation, and prototyping • Data analysis, exploration, and visualization • Scientific and engineering graphics • Application development, including graphical user interface building MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran. The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation. MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The MATLAB System The MATLAB system consists of five main parts: Development Environment. This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path. The MATLAB Mathematical Function Library. This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms. The MATLAB Language. This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both “programming in the small” to rapidly create quick and dirty throw-away programs, and “programming in the large” to create large and complex application programs. Graphics. MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications. The MATLAB External Interfaces/API. This is a library that allows you to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

MATLAB Documentation MATLAB provides extensive documentation, in both printed and online format, to help you learn about and use all of its features. If you are a new user, start with this Getting Started book. It covers all the primary MATLAB features at a high level, including many examples. The MATLAB online help provides task-oriented and reference information about MATLAB features. MATLAB documentation is also available in printed form and in PDF format. MATLAB Online Help To view the online documentation, select MATLAB Help from the Help menu in MATLAB. The MATLAB documentation is organized into these main topics: • Desktop Tools and Development Environment — Startup and shutdown, the desktop, and other tools that help you use MATLAB • Mathematics — Mathematical operations and data analysis • Programming — The MATLAB language and how to develop MATLAB applications • Graphics — Tools and techniques for plotting, graph annotation, printing, and programming with Handle Graphics® • 3-D Visualization — Visualizing surface and volume data, transparency, and viewing and lighting techniques • Creating Graphical User Interfaces — GUI-building tools and how to write callback functions • External Interfaces/API — MEX-files, the MATLAB engine, and interfacing to Java, COM, and the serial port MATLAB also includes reference documentation for all MATLAB functions: • Functions - By Category — Lists all MATLAB functions grouped into categories • Handle Graphics Property Browser — Provides easy access to descriptions of graphics object properties • External Interfaces/API Reference — Covers those functions used by the MATLAB external interfaces, providing information on syntax in the calling language, description, arguments, return values, and examples

Starting and Quitting MATLAB Starting MATLAB On Windows platforms, start MATLAB by double-clicking the MATLAB shortcut icon on your Windows desktop. On UNIX platforms, start MATLAB by typing matlab at the operating system prompt. You can customize MATLAB startup. For example, you can change the directory in which MATLAB starts or automatically execute MATLAB statements in a script file named startup.m.

MATLAB Desktop When you start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB. The following illustration shows the default desktop. You can customize the arrangement of tools and documents to suit your needs. For more information about the desktop tools, see Chapter 6, “Desktop Tools and Development Environment.”

Description: Graphical user interface, application

Description automatically generated

Matrices and Magic Squares In MATLAB, a matrix is a rectangular array of numbers. Special meaning is sometimes attached to 1-by-1 matrices, which are scalars, and to matrices with only one row or column, which are vectors. MATLAB has other ways of storing both numeric and nonnumeric data, but in the beginning, it is usually best to think of everything as a matrix. The operations in MATLAB are designed to be as natural as possible. Where other programming languages work with numbers one at a time, MATLAB allows you to work with entire matrices quickly and easily. A good example matrix, used throughout this book, appears in the Renaissance engraving Melencolia I by the German artist and amateur mathematician Albrecht Dürer.

Description: A picture containing text

Description automatically generated

This image is filled with mathematical symbolism, and if you look carefully, you will see a matrix in the upper right corner. This matrix is known as a magic square and was believed by many in Dürer’s time to have genuinely magical properties. It does turn out to have some fascinating characteristics worth exploring.

Description: A picture containing text, furniture, chest of drawers

Description automatically generated

Entering Matrices The best way for you to get started with MATLAB is to learn how to handle matrices. Start MATLAB and follow along with each example. You can enter matrices into MATLAB in several different ways: • Enter an explicit list of elements. • Load matrices from external data files. • Generate matrices using built-in functions. • Create matrices with your own functions in M-files. Start by entering Dürer’s matrix as a list of its elements. You only have to follow a few basic conventions: • Separate the elements of a row with blanks or commas. • Use a semicolon, ; , to indicate the end of each row. • Surround the entire list of elements with square brackets, [ ].

To enter Dürer’s matrix, simply type in the Command Window A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1] MATLAB displays the matrix you just entered: A = 16 3 2 13 5 10 11 8 9 6 7 12 4 15 14 1 This matrix matches the numbers in the engraving. Once you have entered the matrix, it is automatically remembered in the MATLAB workspace. You can refer to it simply as A. Now that you have A in the workspace, take a look at what makes it so interesting. Why is it magic? sum, transpose, and diag You are probably already aware that the special properties of a magic square have to do with the various ways of summing its elements. If you take the sum along any row or column, or along either of the two main diagonals, you will always get the same number. Let us verify that using MATLAB. The first statement to try is sum(A) MATLAB replies with ans = 34 34 34 34 When you do not specify an output variable, MATLAB uses the variable ans, short for answer, to store the results of a calculation. You have computed a row vector containing the sums of the columns of A. Sure enough, each of the columns has the same sum, the magic sum, 34. How about the row sums? MATLAB has a preference for working with the columns of a matrix, so the easiest way to get the row sums is to transpose the matrix, compute the column sums of the transpose, and then transpose the result. The transpose operation is denoted by an apostrophe or single quote, '. It flips a matrix about its main diagonal and it turns a row vector into a column vector.

The other diagonal, the so-called antidiagonal, is not so important mathematically, so MATLAB does not have a ready-made function for it. But a function originally intended for use in graphics, fliplr, flips a matrix from left to right: sum(diag(fliplr(A))) ans = 34 You have verified that the matrix in Dürer’s engraving is indeed a magic square and, in the process, have sampled a few MATLAB matrix operations. The following sections continue to use this matrix to illustrate additional MATLAB capabilities. Subscripts The element in row i and column j of A is denoted by A(i,j). For example, A(4,2) is the number in the fourth row and second column. For our magic square, A(4,2) is 15. So to compute the sum of the elements in the fourth column of A, type A(1,4) + A(2,4) + A(3,4) + A(4,4) This produces ans = 34 but is not the most elegant way of summing a single column. It is also possible to refer to the elements of a matrix with a single subscript, A(k). This is the usual way of referencing row and column vectors. But it can also apply to a fully two-dimensional matrix, in which case the array is regarded as one long column vector formed from the columns of the original matrix. So, for our magic square, A(8) is another way of referring to the value 15 stored in A(4,2). If you try to use the value of an element outside of the matrix, it is an error: t = A(4,5) Index exceeds matrix dimensions.

Graph Components MATLAB displays graphs in a special window known as a figure. To create a graph, you need to define a coordinate system. Therefore every graph is placed within axes, which are contained by the figure. The actual visual representation of the data is achieved with graphics objects like lines and surfaces. These objects are drawn within the coordinate system defined by the axes, which MATLAB automatically creates specifically to accommodate the range of the data. The actual data is stored as properties of the graphics objects. See “Handle Graphics” on page 3-62 for more information about graphics object properties. The following picture shows the basic components of a typical graph.

Description: Graphical user interface

Description automatically generated

Figure Tools The figure is equipped with sets of tools that operate on graphs. The figure Tools menu provides access to many graph tools.

Description: Graphical user interface, application, Word

Description automatically generated

Access to Tools You can access the figure toolbars and the plotting tools from the View menu, as shown in the following picture.

Description: Graphical user interface, application, Word

Description automatically generated

Editor/Debugger Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for text editing, as well as for M-file debugging. To create or edit an M-file use File -> New or File -> Open, or use the edit function.

Description: Graphical user interface, text, application, email

Description automatically generated

You can use any text editor to create M-files, such as Emacs. Use preferences (accessible from the desktop File menu) to specify that editor as the default. If you use another editor, you can still use the MATLAB Editor/Debugger for debugging, or you can use debugging functions, such as dbstop, which sets a breakpoint. If you just need to view the contents of an M-file, you can display the contents in the Command Window using the type function.

**CHAPTER 5**

**Materials and methods**

In this chapter, the datasets used for training and testing and the deep learning model used in this study will be discussed. Datasets are discussed further in Section 5.1. Likewise, Section 5.2 discusses the proposed classification method for fundus images.

**Proposed method**

The proposed fundus image classification model based on attention mechanism and feature fusion which is named CNN-Trans model in this paper. Its architecture is shown in Fig. 3. The model is a parallel dualbranch network. Firstly, the patient’s fundus image is cropped to remove the black area, and the size is reset to 224 × 224 × 3, and then normalized. After the above preprocessing steps are completed, it is ready to be sent to the next model. Then, on the one hand, the preprocessed pictures enter the CNNLSTM model branch. Firstly, a feature map with a size of 7 × 7 × 2048 is extracted through pre-trained Xception, and then the coordinated attention module decomposes channel attention into two onedimensional feature encoding processes, and aggregates features along two spatial directions to obtain a feature map. Then, the feature maps are encoded as a pair of orientation-aware pairs and a location-sensitive attention map to augment the representation of objects of interest, these steps can effectively improve the processing efficiency of the input feature maps. After this, the feature map of the coordinated attention

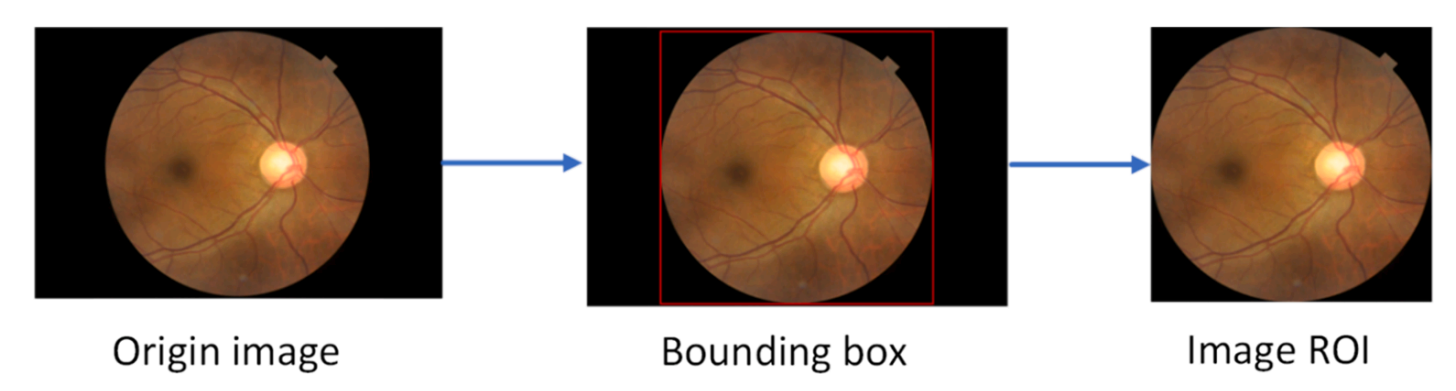


Fig. 2. process of removing black areas.

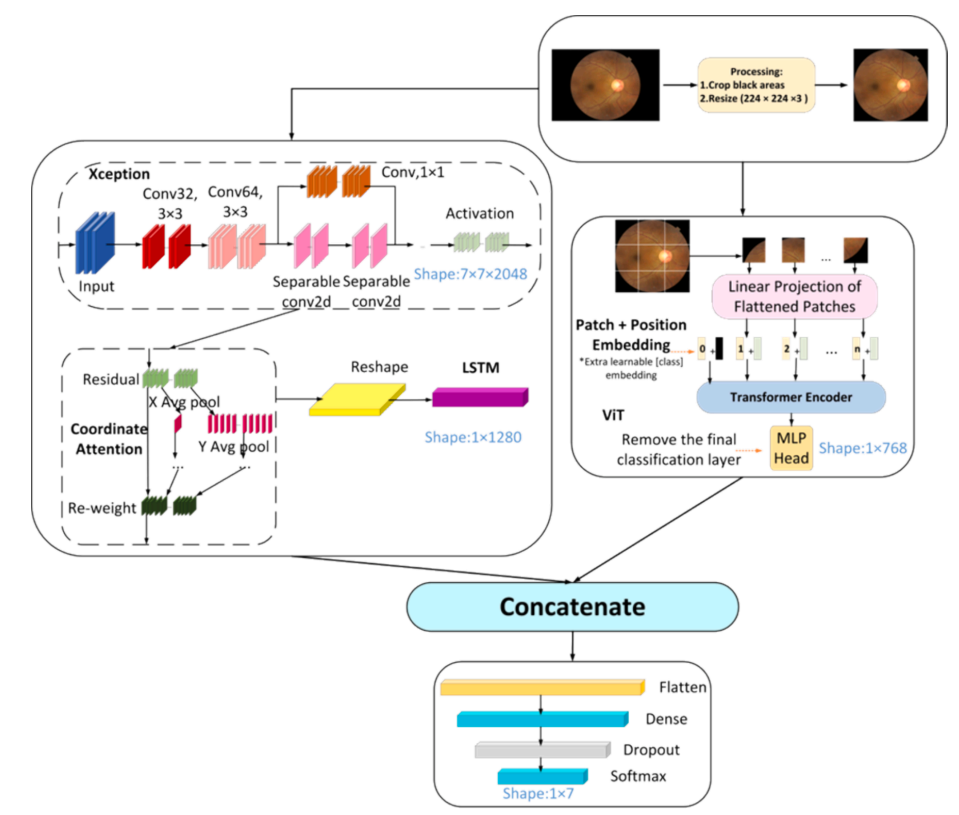


Fig. 3. The overall architecture of the CNN-Trans model.

module is input to the reshape layer, which changes the shape from (h, w, c) to (h \* w, c) input to LSTM. On the other hand, the preprocessed image enters the ViT branch. First, the input image is divided into image blocks and serialized, position encoding is added, then adds learnable embedding vectors, and inputs them into the encoder for encoding, and finally outputs Learnable embedding vectors are used for subsequent classification. Finally, the features extracted by the Xception branch and the features extracted by the ViT branch are concat spliced and input to the reconstructed classification head, that is, a Flatten layer, a Dense layer with an L2 regularization parameter of 0.01 and 512 output units, and a Dropout layer with a dropout rate of 0.5, a Dense layer with a SoftMax function and an output unit of 7. 3.2.1. CNN-LSTM branch The CNN-LSTM branch mainly consists of three parts: Xception, LSTM and coordinated attention mechanism, which is also part of our previous research work [18]. (1) Xception The limited available data discourages the use of large CNN models, and very large models such as DenseNet-121 may negatively impact generalization ability [19]. Therefore, when we choose the model, we consider the lightweight and simple model as much as possible, and use the model with better initial classification effect as the feature extractor through simple experiments. In the end, we chose Xception as the original feature extractor: Xception [20] is an improvement to Inception-v3 proposed by Google after Inception in 2016. The structure is shown in Fig. 4. Its full name is “Extreme Inception”, that is, Limit Inception. Xception uses depthwise separable convolution (Depthwise Separable Convolution) instead of conventional convolution operations, which is different from Inception. Depth-separable convolution divides the standard convolution operation into two steps: depthwise convolution and pointwise convolution. The former focuses on the spatial features within the input channel, while the latter focuses on the relationship between different channels. In this way, the number of parameters and calculations in the model can be greatly reduced, the computational efficiency of the model can be improved and the risk of overfitting can be reduced. In addition, Xception also introduces a special extreme Inception block, which replaces the standard convolution operation in the branch convolutional layer of the Inception block with a depthwise separable convolution. The role of this extreme Inception block is to further reduce the number of parameters and calculations in the model and improve the accuracy of the model. Xception has achieved very good performance in many computer vision fields, such as image classification, target detection, semantic segmentation and other tasks. In the ImageNet image classification challenge, Xception achieved an accuracy of 0.790 and 0.945 on the Top-1 and Top-5 error rates, respectively, surpassing the most advanced ResNet model at the time. (2) LSTM. Long Short-Term Memory (LSTM) is a common Recurrent Neural Network (RNN) architecture. The emergence of LSTM is mainly due to the problem of gradient disappearance or explosion in traditional RNN

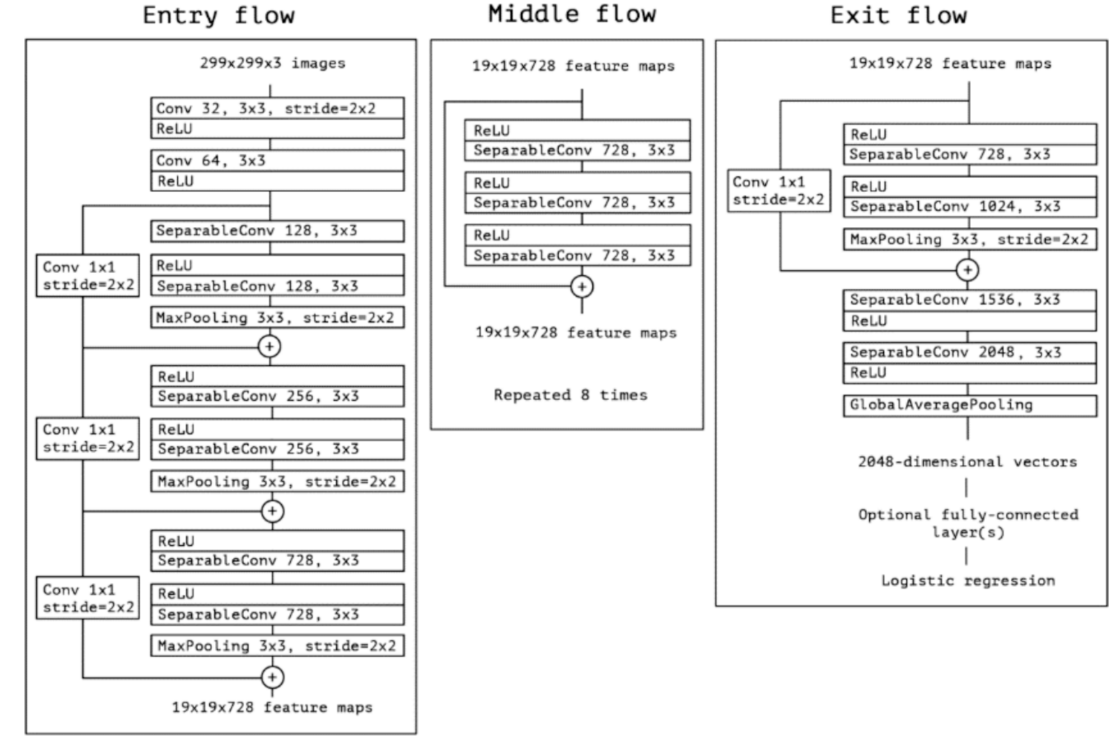
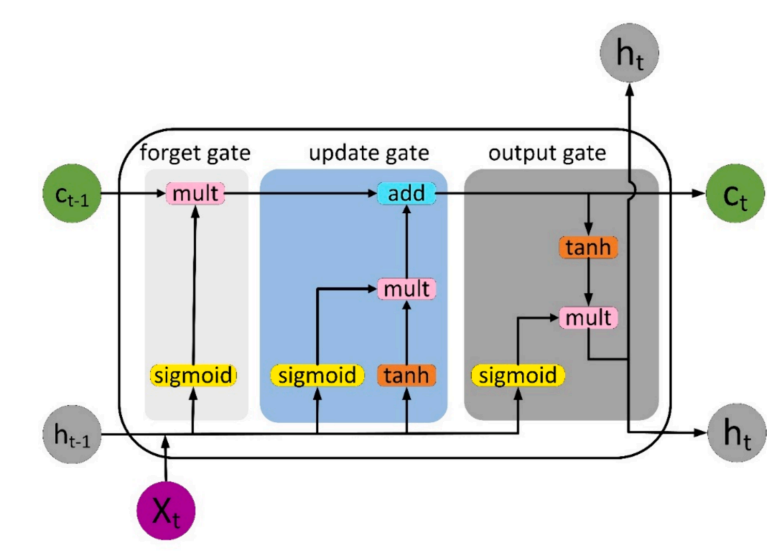


Fig Architecture

It is a RNN based on a gating mechanism. The structure of LSTM consists of a group of special neuronal units that can selectively forget, retain or add information to achieve long-term memory. The basic units of LSTM include forget gate, input gate and output gate, which fuse the input data and the information of the previous state through nonlinear transformation, and decide whether to retain or



forget certain information through the gating mechanism. The forget gate controls the forgetting of the past state, the input gate controls the input of new information, and the output gate controls the output of the current state. During the training process of LSTM, these gates are trained to automatically adapt to the pattern of input data and the corresponding context information. The value range of the gate is (0,1), controlled by the Sigmoid function, where Xt refers to the current input, ct and ct-1 denote the new and previous cell states, respectively, ht and ht1 are the current and previous outputs, respectively. The internal structure of LSTM is shown in Fig. 5. LSTM adds a cell state to save the long-term state, which is its main difference from RNN, which can remember the previous information and connect it to the currently obtained data. LSTM is mainly used to process sequence data, which can capture the long-term dependence in the sequence, while CNN is mainly used to process image data, which can capture the spatial local features in the image. Combining the two and making full use of their respective characteristics can effectively improve the training speed and efficiency of the model, and maintain the state information of the features encountered in the previous generation of image classification. (3) Coordinate Attention Coordinate Attention (CoordAtt for short) is a plug-in proposed by Qibin Hou et al. [22] inspired by the Squeeze-Excitation (SE) attention module in 2021. A ready-to-use mobile network attention mechanism, coordinated attention is different from channel attention, it uses two one-dimensional feature encoding processes to aggregate features in different spatial directions to capture long-range dependencies and precise location information at the same time, and ultimately achieve enhanced object of interest expressed purpose. Its structure is shown in Fig. 6. Qibin Hou et al. found that compared with the most popular SE attention module in mobile networks, the coordinated attention mechanism is simple, can be flexibly inserted into the network, and has almost no computational overhead. While taking advantage of the modularity, it is also able to capture the long-term dependencies of precise position information.

**5.2.2. ViT branch** Vision Transformer (ViT for short) is a method of using the Transformer model to process image data [23], proposed by Alexey Dosovitskiy et al. in 2020. The core idea of ViT is to divide the image into several small blocks, then convert these small blocks into vectors, and input these vectors into the Transformer model for processing. The ViT architecture is shown in Fig. 7, which mainly consists of a module that splits the input image into blocks, a layer that embeds blocks and positions, a Transformer encoder, and an MLP classification head. Specifically, ViT first divides the image into some small blocks of the same size, and then maps each small block into a vector through a fully connected layer. These vectors are treated as tokens in the sequence and fed into the Transformer encoder for processing. In the Transformer encoder, each token is encoded through a multi-layer self-attention mechanism and a feed-forward neural network, and a contextdependent vector representation is obtained. Finally, these vectors are passed through a fully connected layer for classification or regression. In this way, ViT does not need to use traditional convolutional neural networks, and can perform tasks such as classification and detection on images using only Transformer, and has achieved good results on some standard computer vision datasets. When training on large amounts of data in the early stage, the performance of ViT is also comparable to the performance of CNN on small or medium-sized datasets [24].

**5.2.3. Feature fusion**

In the feature fusion of image processing, the two commonly used feature fusion (FLF) methods are concatand add. addThe method increases the information content of the image features, but does not increase the dimensionality of the image. On the other hand, concatthe method merges the number of channels to increase the number of channels describing the image, thereby keeping the relevant information of each feature unchanged. If xthe dimension of the sum of the two input features is p, ythen qthe zdimension of the output feature is p + q, as shown in Fig. 8. Equations (2) and (3) are related mathematical expressions. In this study, since the number of channels for extracting features from the two parallel branches of ViT and Xception is different, we use concat to fuse the features extracted by the two. as the picture shows. Fusion( xy ) The number of channels is Feature(x) the sum of the number of channels of and Feature(y).

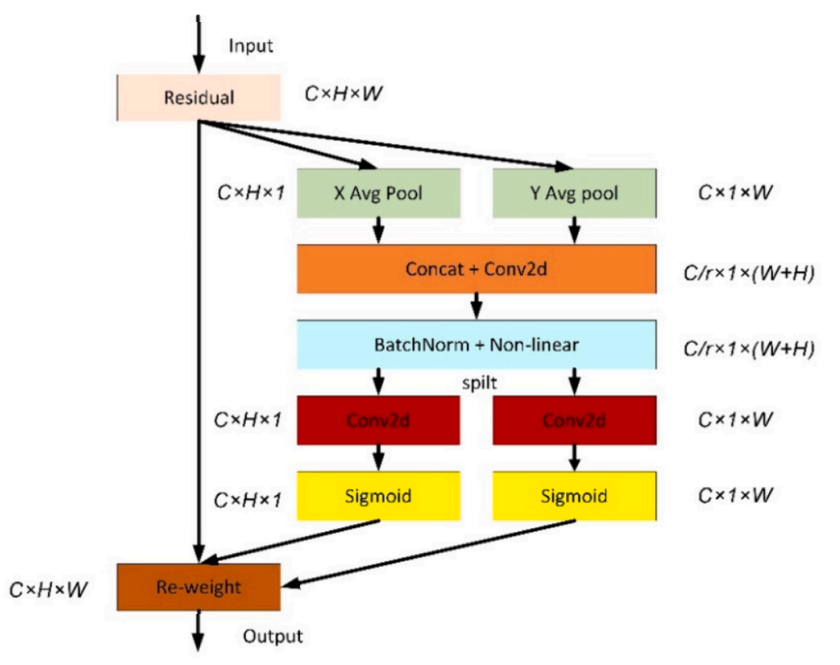
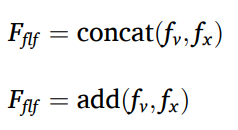


Fig. 6. The structure of coordinated attention.



Among them, fv the features extracted fx by ViT are the features extracted by Fflf Xception, which is the fused feature set. Concat fusion in the CNN-Trans model is: in the branch CNN-LSTM and another branch ViT, they can both perform feature extraction on the input medical image data and output a vector representation. These two vector representations can represent different understandings of input data by CNN-LSTM and ViT respectively. In order to fuse the different understandings extracted by the two, that is, the features of medical images, and use the fused results for final classification. At this point, this article uses the concat operation to splice these two vectors along a certain dimension, and the resulting new vector will be a tensor of shape (N, c1 + c2), where N represents the number of samples, and c1 and c2 are respectively Represents the vector length of CNN-LSTM and ViT output. This new fused tensor contains the features of CNN-LSTM and ViT, is a richer vector representation, and is used for the final classification task. It should be noted that the concat fusion method needs to normalize or scale the feature representation output by each model before splicing to avoid large deviations in the spliced results. At the same time, the concat fusion method also has some shortcomings, such as increasing the amount of calculation and storage of the model, so it needs to be selected and optimized according to specific tasks and data conditions.

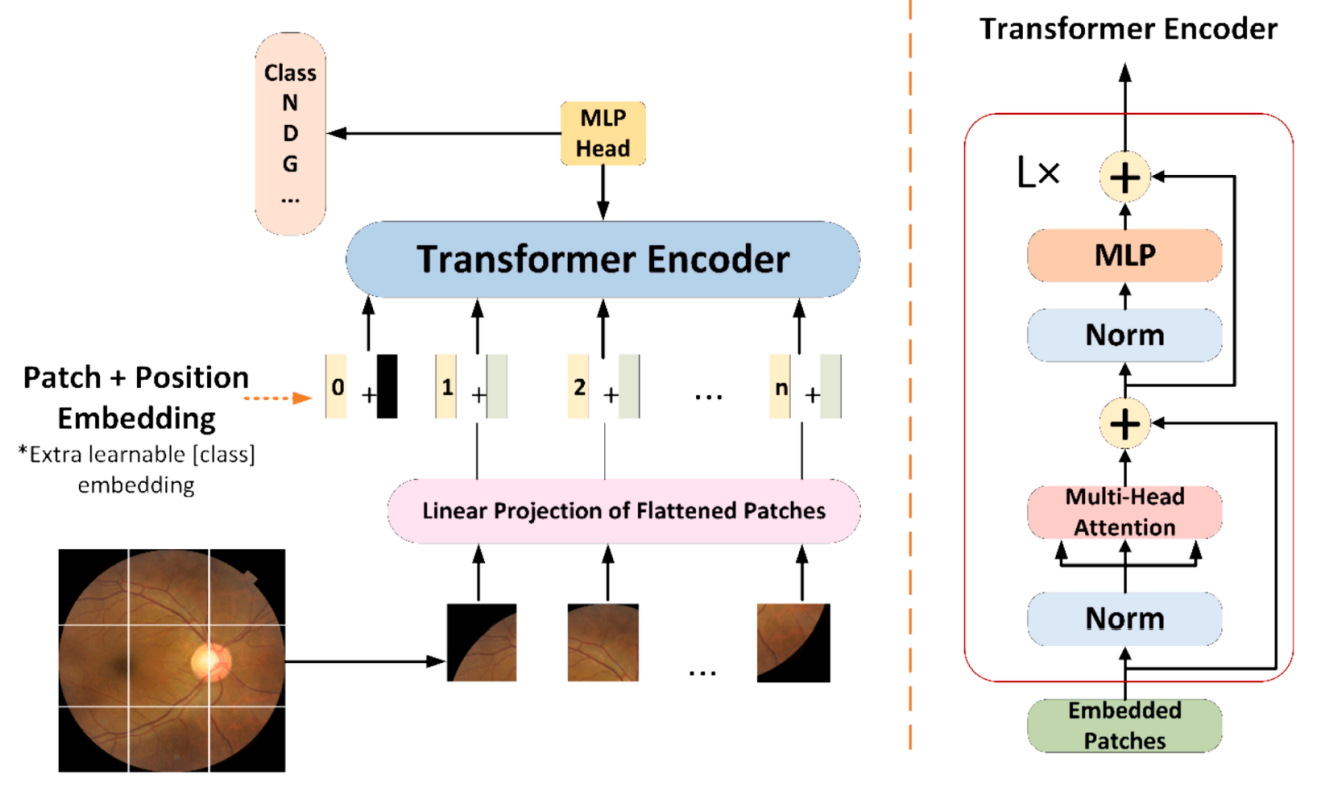


Fig. 7. The architecture of ViT

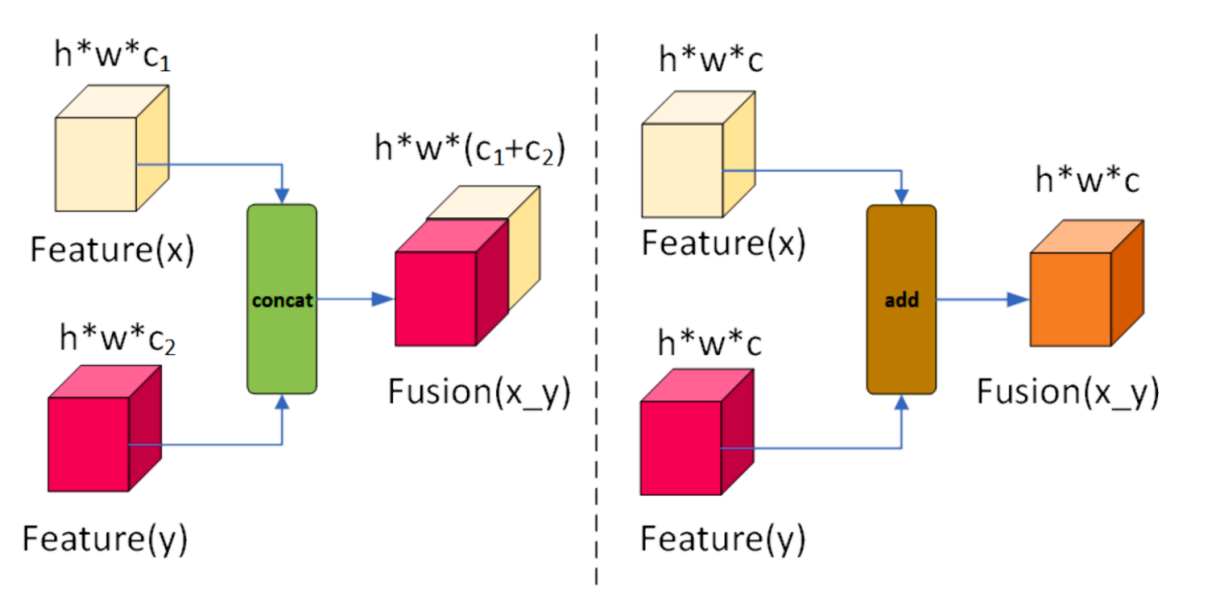


Fig. 8. Two of the most commonly used feature fusion methods

**CHAPTER 6**

**Evaluation indicators**

For each image, if the predicted value of a certain label is greater than 0.5, it will be identified as a hard label in the experiment. According to the predicted label and the real label, the confusion matrix of each category can be obtained. Performance indicators based on confusion matrix calculations, such as Precision, Sensitivity (Recall), Specificity, F1, and Accuracy are still essential:

Precision = TP TP + FP (4)

Sensitivity(Recall) = TP TP + FN (5)

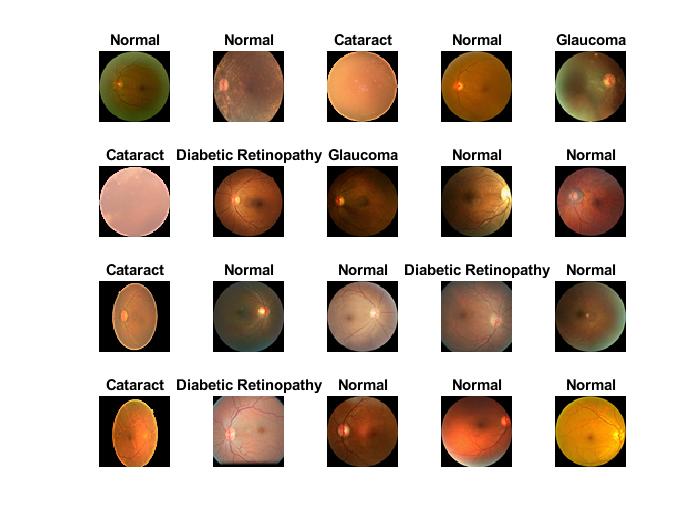
Specificity = TN TN + FP (6)

F1 − score = 2 × Precision × Recall Precision + Recall (7)

Accuracy = TP + TN TP + FN + TN + FP (8) TP, TN, FP, and FN are elements of the confusion matrix (see Table 3).

>> MAIN

In MAIN (line 58)



Training on single GPU.

Initializing input data normalization.

|==================================================================================|

| Epoch | Iteration | Time Elapsed | Mini-batch | Validation | Mini-batch | Validation | Base Learning |

| | | (hh:mm:ss) | Accuracy | Accuracy | Loss | Loss | Rate |

|==================================================================================|

| 1 | 1 | 00:00:44 | 28.12% | 29.86% | 1.5182 | 1.5324 | 1.0000e-04 |

| 1 | 20 | 00:00:52 | 34.38% | 57.35% | 1.5335 | 0.9833 | 1.0000e-04 |

| 1 | 40 | 00:00:56 | 59.38% | 67.06% | 0.7751 | 0.8098 | 1.0000e-04 |

| 1 | 50 | 00:01:01 | 68.75% | | 0.8914 | | 1.0000e-04 |

| 1 | 60 | 00:01:09 | 81.25% | 73.22% | 0.7276 | 0.7198 | 1.0000e-04 |

| 1 | 80 | 00:01:15 | 65.62% | 72.99% | 0.6844 | 0.6580 | 1.0000e-04 |

| 1 | 100 | 00:01:24 | 68.75% | 75.12% | 0.7174 | 0.6199 | 1.0000e-04 |

| 2 | 120 | 00:01:29 | 90.62% | 80.33% | 0.4019 | 0.5706 | 1.0000e-04 |

| 2 | 140 | 00:01:33 | 84.38% | 80.81% | 0.4104 | 0.5534 | 1.0000e-04 |

| 2 | 150 | 00:01:37 | 75.00% | | 0.5480 | | 1.0000e-04 |

| 2 | 160 | 00:01:40 | 81.25% | 82.46% | 0.4915 | 0.5163 | 1.0000e-04 |

| 2 | 180 | 00:01:45 | 81.25% | 82.23% | 0.3252 | 0.4963 | 1.0000e-04 |

| 2 | 200 | 00:01:50 | 71.88% | 84.12% | 0.5894 | 0.4771 | 1.0000e-04 |

| 2 | 220 | 00:01:58 | 93.75% | 84.12% | 0.4480 | 0.4662 | 1.0000e-04 |

| 3 | 240 | 00:02:02 | 78.12% | 84.12% | 0.4480 | 0.4448 | 1.0000e-04 |

| 3 | 250 | 00:02:03 | 81.25% | | 0.4658 | | 1.0000e-04 |

| 3 | 260 | 00:02:05 | 71.88% | 87.20% | 0.5561 | 0.4423 | 1.0000e-04 |

| 3 | 280 | 00:02:11 | 81.25% | 85.78% | 0.4662 | 0.4214 | 1.0000e-04 |

| 3 | 300 | 00:02:14 | 78.12% | 85.55% | 0.5315 | 0.4116 | 1.0000e-04 |

| 3 | 320 | 00:02:16 | 93.75% | 87.20% | 0.3317 | 0.4029 | 1.0000e-04 |

| 3 | 340 | 00:02:20 | 93.75% | 88.63% | 0.2930 | 0.3926 | 1.0000e-04 |

| 3 | 350 | 00:02:21 | 84.38% | | 0.5097 | | 1.0000e-04 |

| 4 | 360 | 00:02:23 | 87.50% | 86.02% | 0.3438 | 0.3837 | 1.0000e-04 |

| 4 | 380 | 00:02:26 | 84.38% | 87.44% | 0.3415 | 0.3814 | 1.0000e-04 |

| 4 | 400 | 00:02:29 | 84.38% | 88.39% | 0.3314 | 0.3698 | 1.0000e-04 |

| 4 | 420 | 00:02:32 | 90.62% | 87.68% | 0.3447 | 0.3630 | 1.0000e-04 |

| 4 | 440 | 00:02:35 | 96.88% | 89.81% | 0.1908 | 0.3563 | 1.0000e-04 |

| 4 | 450 | 00:02:36 | 90.62% | | 0.2711 | | 1.0000e-04 |

| 4 | 460 | 00:02:37 | 90.62% | 90.28% | 0.2701 | 0.3486 | 1.0000e-04 |

| 5 | 480 | 00:02:40 | 93.75% | 87.44% | 0.2890 | 0.3489 | 1.0000e-04 |

| 5 | 500 | 00:02:44 | 84.38% | 86.73% | 0.3613 | 0.3547 | 1.0000e-04 |

| 5 | 520 | 00:02:48 | 93.75% | 90.05% | 0.2343 | 0.3356 | 1.0000e-04 |

| 5 | 540 | 00:02:51 | 81.25% | 89.57% | 0.2844 | 0.3263 | 1.0000e-04 |

| 5 | 550 | 00:02:52 | 87.50% | | 0.4144 | | 1.0000e-04 |

| 5 | 560 | 00:02:53 | 84.38% | 90.28% | 0.3041 | 0.3252 | 1.0000e-04 |

| 5 | 580 | 00:02:56 | 90.62% | 90.05% | 0.2719 | 0.3245 | 1.0000e-04 |

| 6 | 600 | 00:02:59 | 93.75% | 88.63% | 0.2478 | 0.3276 | 1.0000e-04 |

| 6 | 620 | 00:03:02 | 81.25% | 89.10% | 0.3729 | 0.3279 | 1.0000e-04 |

| 6 | 640 | 00:03:05 | 84.38% | 90.28% | 0.3746 | 0.3102 | 1.0000e-04 |

| 6 | 650 | 00:03:06 | 87.50% | | 0.2246 | | 1.0000e-04 |

| 6 | 660 | 00:03:08 | 87.50% | 90.76% | 0.3004 | 0.3045 | 1.0000e-04 |

| 6 | 680 | 00:03:11 | 87.50% | 91.00% | 0.2504 | 0.3028 | 1.0000e-04 |

| 6 | 700 | 00:03:13 | 96.88% | 88.39% | 0.2079 | 0.3143 | 1.0000e-04 |

| 7 | 720 | 00:03:16 | 90.62% | 89.10% | 0.2532 | 0.3111 | 1.0000e-04 |

| 7 | 740 | 00:03:19 | 87.50% | 90.52% | 0.3532 | 0.2948 | 1.0000e-04 |

| 7 | 750 | 00:03:20 | 84.38% | | 0.2618 | | 1.0000e-04 |

| 7 | 760 | 00:03:22 | 100.00% | 91.00% | 0.1778 | 0.2928 | 1.0000e-04 |

| 7 | 780 | 00:03:25 | 90.62% | 91.00% | 0.2647 | 0.2901 | 1.0000e-04 |

| 7 | 800 | 00:03:27 | 93.75% | 91.23% | 0.1978 | 0.2850 | 1.0000e-04 |

| 7 | 820 | 00:03:30 | 78.12% | 89.10% | 0.6030 | 0.2976 | 1.0000e-04 |

| 8 | 840 | 00:03:34 | 93.75% | 90.52% | 0.1989 | 0.2898 | 1.0000e-04 |

| 8 | 850 | 00:03:35 | 84.38% | | 0.3589 | | 1.0000e-04 |

| 8 | 860 | 00:03:37 | 93.75% | 91.23% | 0.1936 | 0.2762 | 1.0000e-04 |

| 8 | 880 | 00:03:41 | 90.62% | 91.23% | 0.3422 | 0.2766 | 1.0000e-04 |

| 8 | 900 | 00:03:44 | 93.75% | 90.52% | 0.1707 | 0.2773 | 1.0000e-04 |

| 8 | 920 | 00:03:47 | 90.62% | 91.23% | 0.3193 | 0.2721 | 1.0000e-04 |

| 8 | 940 | 00:03:50 | 87.50% | 90.52% | 0.3610 | 0.2743 | 1.0000e-04 |

| 9 | 950 | 00:03:51 | 93.75% | | 0.2371 | | 1.0000e-04 |

| 9 | 960 | 00:03:53 | 90.62% | 89.81% | 0.2849 | 0.2722 | 1.0000e-04 |

| 9 | 980 | 00:03:56 | 93.75% | 92.42% | 0.1248 | 0.2669 | 1.0000e-04 |

| 9 | 1000 | 00:03:58 | 96.88% | 90.76% | 0.1268 | 0.2668 | 1.0000e-04 |

| 9 | 1020 | 00:04:02 | 96.88% | 90.28% | 0.1706 | 0.2660 | 1.0000e-04 |

| 9 | 1040 | 00:04:06 | 93.75% | 91.00% | 0.2057 | 0.2635 | 1.0000e-04 |

| 9 | 1050 | 00:04:07 | 96.88% | | 0.1820 | | 1.0000e-04 |

| 9 | 1060 | 00:04:08 | 81.25% | 90.05% | 0.2982 | 0.2633 | 1.0000e-04 |

| 10 | 1080 | 00:04:11 | 90.62% | 90.05% | 0.1564 | 0.2633 | 1.0000e-04 |

| 10 | 1100 | 00:04:14 | 96.88% | 91.94% | 0.1667 | 0.2595 | 1.0000e-04 |

| 10 | 1120 | 00:04:17 | 96.88% | 90.52% | 0.1732 | 0.2641 | 1.0000e-04 |

| 10 | 1140 | 00:04:20 | 90.62% | 91.00% | 0.2912 | 0.2574 | 1.0000e-04 |

| 10 | 1150 | 00:04:21 | 90.62% | | 0.2083 | | 1.0000e-04 |

| 10 | 1160 | 00:04:23 | 90.62% | 90.05% | 0.2122 | 0.2616 | 1.0000e-04 |

| 10 | 1180 | 00:04:25 | 90.62% | 89.81% | 0.2363 | 0.2704 | 1.0000e-04 |

| 11 | 1200 | 00:04:28 | 93.75% | 91.47% | 0.1659 | 0.2637 | 1.0000e-04 |

| 11 | 1220 | 00:04:31 | 100.00% | 92.65% | 0.1419 | 0.2498 | 1.0000e-04 |

| 11 | 1240 | 00:04:34 | 100.00% | 90.28% | 0.1108 | 0.2666 | 1.0000e-04 |

| 11 | 1250 | 00:04:35 | 90.62% | | 0.2213 | | 1.0000e-04 |

| 11 | 1260 | 00:04:37 | 100.00% | 91.94% | 0.1532 | 0.2498 | 1.0000e-04 |

| 11 | 1280 | 00:04:39 | 87.50% | 90.52% | 0.2752 | 0.2541 | 1.0000e-04 |

| 12 | 1300 | 00:04:42 | 90.62% | 89.81% | 0.1653 | 0.2711 | 1.0000e-04 |

| 12 | 1320 | 00:04:45 | 96.88% | 91.23% | 0.1440 | 0.2582 | 1.0000e-04 |

| 12 | 1340 | 00:04:48 | 90.62% | 91.47% | 0.2052 | 0.2433 | 1.0000e-04 |

| 12 | 1350 | 00:04:49 | 100.00% | | 0.1243 | | 1.0000e-04 |

| 12 | 1360 | 00:04:51 | 96.88% | 89.57% | 0.1020 | 0.2655 | 1.0000e-04 |

| 12 | 1380 | 00:04:54 | 93.75% | 91.47% | 0.2048 | 0.2432 | 1.0000e-04 |

| 12 | 1400 | 00:04:57 | 100.00% | 90.28% | 0.1258 | 0.2474 | 1.0000e-04 |

| 13 | 1420 | 00:04:59 | 96.88% | 90.52% | 0.1807 | 0.2520 | 1.0000e-04 |

| 13 | 1440 | 00:05:02 | 84.38% | 90.05% | 0.3007 | 0.2476 | 1.0000e-04 |

| 13 | 1450 | 00:05:03 | 96.88% | | 0.1169 | | 1.0000e-04 |

| 13 | 1460 | 00:05:05 | 93.75% | 91.71% | 0.2170 | 0.2384 | 1.0000e-04 |

| 13 | 1480 | 00:05:07 | 90.62% | 89.34% | 0.2670 | 0.2637 | 1.0000e-04 |

| 13 | 1500 | 00:05:10 | 96.88% | 91.94% | 0.1209 | 0.2377 | 1.0000e-04 |

| 13 | 1520 | 00:05:13 | 100.00% | 91.47% | 0.1248 | 0.2393 | 1.0000e-04 |

| 14 | 1540 | 00:05:15 | 93.75% | 90.52% | 0.1938 | 0.2392 | 1.0000e-04 |

| 14 | 1550 | 00:05:17 | 93.75% | | 0.2206 | | 1.0000e-04 |

| 14 | 1560 | 00:05:18 | 100.00% | 91.23% | 0.1263 | 0.2399 | 1.0000e-04 |

| 14 | 1580 | 00:05:21 | 93.75% | 91.94% | 0.1361 | 0.2349 | 1.0000e-04 |

| 14 | 1600 | 00:05:24 | 93.75% | 89.81% | 0.2235 | 0.2536 | 1.0000e-04 |

| 14 | 1620 | 00:05:27 | 100.00% | 91.71% | 0.0703 | 0.2326 | 1.0000e-04 |

| 14 | 1640 | 00:05:29 | 96.88% | 91.47% | 0.1412 | 0.2336 | 1.0000e-04 |

| 14 | 1650 | 00:05:30 | 84.38% | | 0.2676 | | 1.0000e-04 |

| 15 | 1660 | 00:05:32 | 96.88% | 90.28% | 0.1620 | 0.2388 | 1.0000e-04 |

| 15 | 1680 | 00:05:35 | 96.88% | 91.23% | 0.1411 | 0.2416 | 1.0000e-04 |

| 15 | 1700 | 00:05:38 | 96.88% | 91.94% | 0.1175 | 0.2318 | 1.0000e-04 |

| 15 | 1720 | 00:05:40 | 87.50% | 90.76% | 0.2196 | 0.2368 | 1.0000e-04 |

| 15 | 1740 | 00:05:43 | 90.62% | 92.18% | 0.1625 | 0.2291 | 1.0000e-04 |

| 15 | 1750 | 00:05:44 | 90.62% | | 0.1636 | | 1.0000e-04 |

| 15 | 1760 | 00:05:46 | 96.88% | 91.00% | 0.1232 | 0.2360 | 1.0000e-04 |

| 15 | 1770 | 00:05:48 | 93.75% | 90.28% | 0.1892 | 0.2538 | 1.0000e-04 |

|=================================================================================|

Validation Accuracy: 0.90284

Precision per class:

0.9904

0.9909

0.8713

0.7570

Recall per class:

0.9810

0.9909

0.7719

0.8710

F1-score per class:

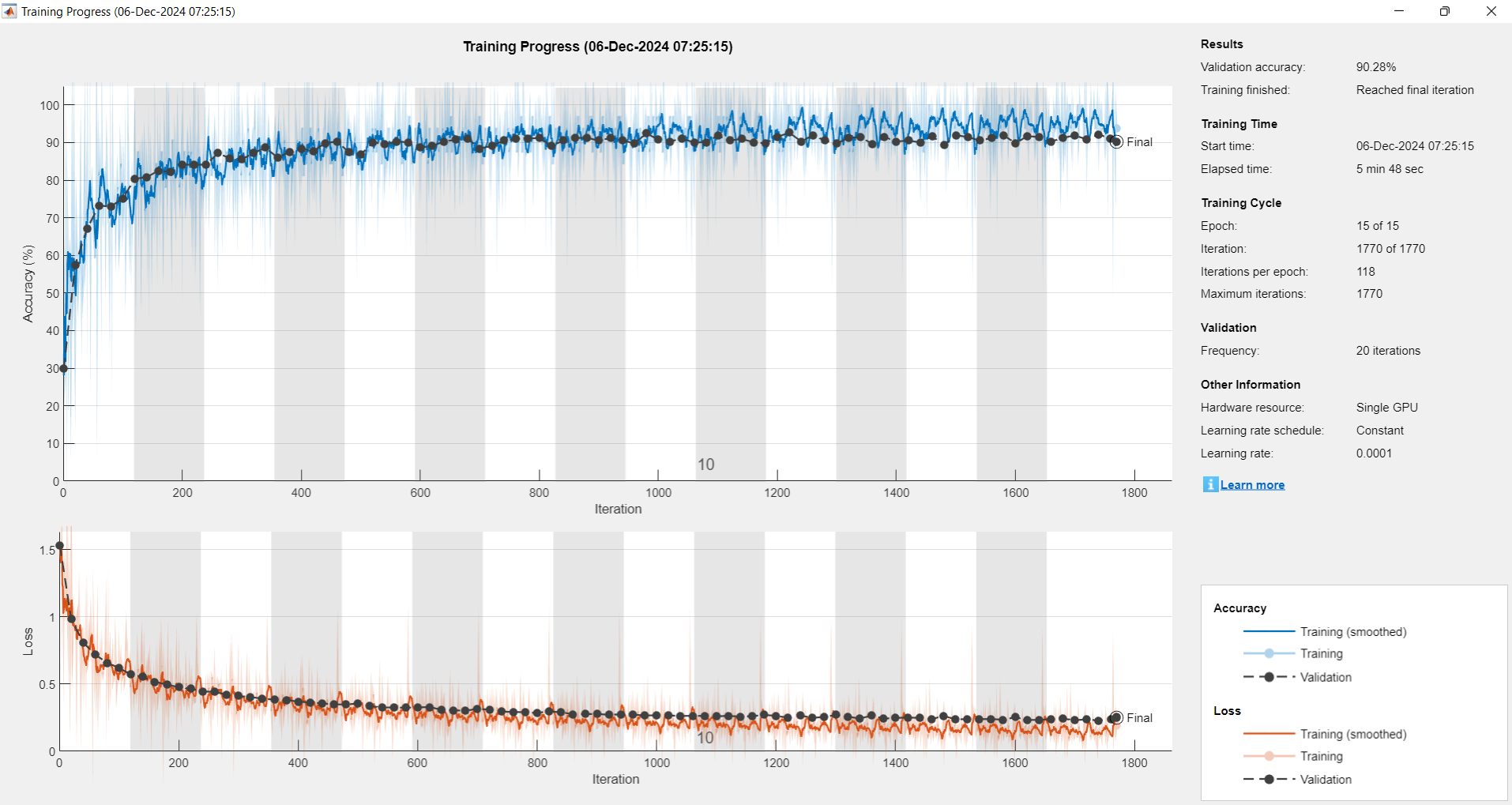
0.9856

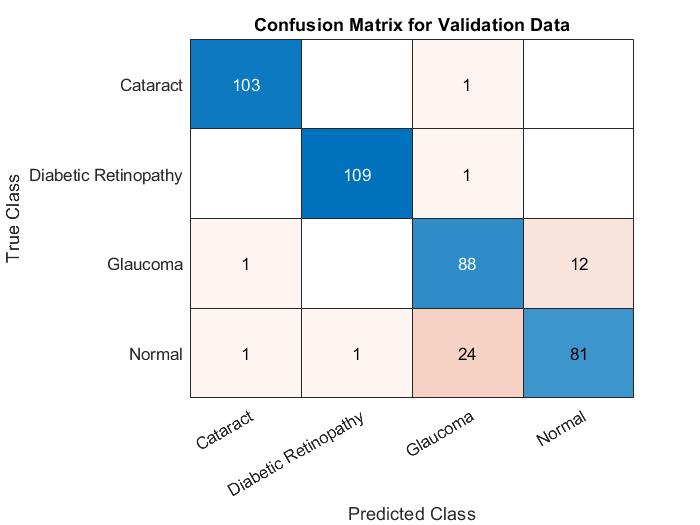
0.9909

0.8186

0.8100

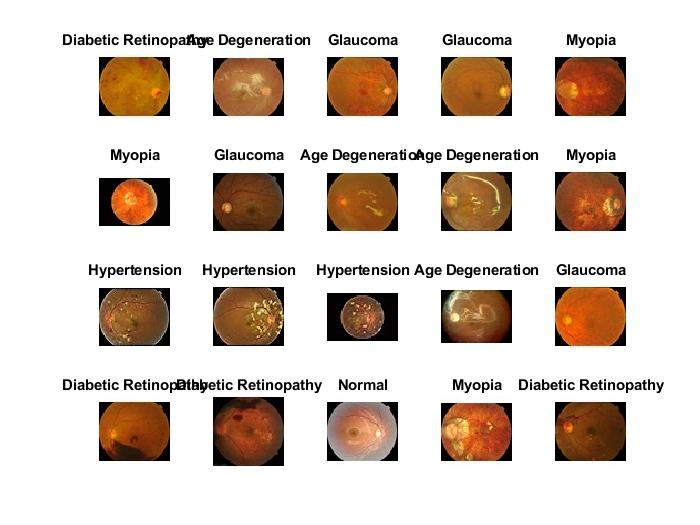
Results saved to classification\_results.mat





7CLASSES

>> MAIN7



Training on single GPU.

Initializing input data normalization.

|======================================================================================================================|

| Epoch | Iteration | Time Elapsed | Mini-batch | Validation | Mini-batch | Validation | Base Learning |

| | | (hh:mm:ss) | Accuracy | Accuracy | Loss | Loss | Rate |

|======================================================================================================================|

| 1 | 1 | 00:00:09 | 3.12% | 13.21% | 2.4804 | 2.3288 | 1.0000e-04 |

| 5 | 30 | 00:00:16 | 90.62% | 84.91% | 0.4805 | 0.6700 | 1.0000e-04 |

| 9 | 50 | 00:00:20 | 93.75% | | 0.3099 | | 1.0000e-04 |

| 10 | 60 | 00:00:22 | 100.00% | 88.68% | 0.1900 | 0.4154 | 1.0000e-04 |

|======================================================================================================================|

Validation Accuracy: 0.88679

Precision per class:

1.0000

0.7500

0.8750

0.7500

1.0000

0.8182

1.0000

Recall per class:

1.0000

1.0000

0.7778

0.6667

1.0000

0.9000

1.0000

F1-score per class:

1.0000

0.8571

0.8235

0.7059

1.0000

0.8571

1.0000

Results saved to classification\_results\_7\_classes.mat

AUC for class Age Degeneration: 1

AUC for class Cataract: 0.875

AUC for class Diabetic Retinopathy: 0.91528

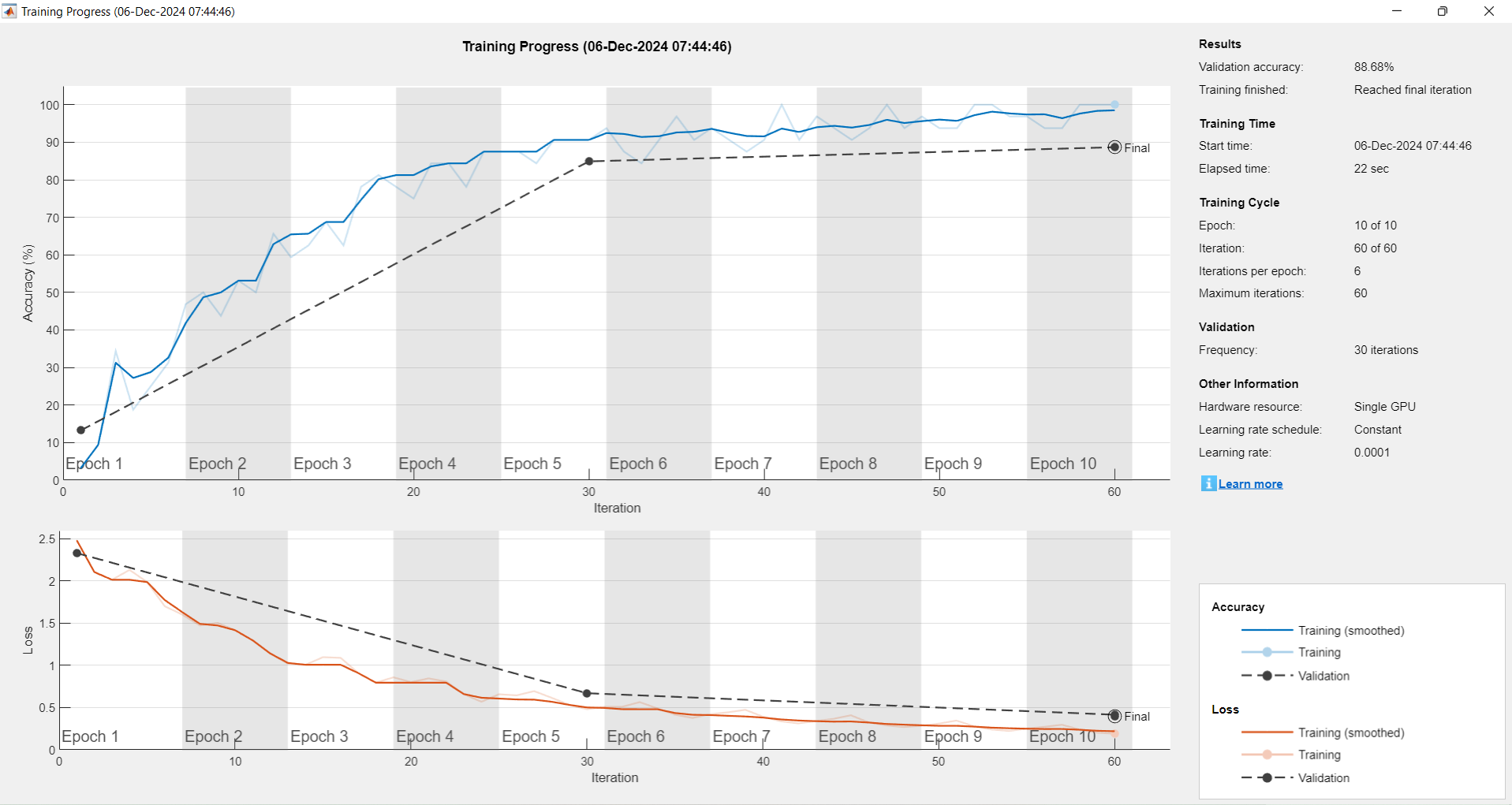
AUC for class Glaucoma: 0.84167

AUC for class Hypertension: 1

AUC for class Myopia: 0.89719

AUC for class Normal: 1

>>





In addition, this paper also added the area under the receiver operating characteristic curve (Receiver Operating Characteristic Curve, referred to as ROC) (Area Under the Curve, referred to as AUC) as a measurement index, the larger the area, the better the performance. The abscissa of the ROC curve is the false positive rate (False positive rate, FPR for short), and the ordinate is the true positive rate (True positive rate, TPR for short), as shown in Fig. 9, the drawing process is as follows:

a. Sort the prediction results according to the predicted positive class probability value;

b. the threshold from 1, and predict the samples as positive examples one by one in this order, and calculate the current FPR and TPR values each time; c. The image is drawn with TPR as the ordinate and FPR as the abscissa. TPR (True Positive Rate) and FPR (False Positive Rate) are calculated as follows: TPR = TP /TP + FN (9) FPR = FP/ FP + TN.

The model is implemented in Keras based on Tensorflow 2.5, and Xception is pre-trained on the ImageNet dataset. First sample 10 % of the data from the training set as the validation set, and then use the Adam optimizer on the prepared data set. For end-to-end retraining, the learning rate is 2 × 10− 4 , the batch size is 32, the epoch value is 50, and the loss selects categorical\_crossentropy. For training, data shuffling is enabled, which includes shuffling the data before each epoch.

Considering that the fundus disease recognition task is a multicategory problem with highly imbalanced categories, it is not suitable to train the model with traditional loss functions. We can use X = x1, x2⋯xn, which is related xi to the real label yi. We wish to find a classification function F : X→Y that minimizes the loss function L, using N sets of labeled training data ( xi, yi ) , where i = 1,⋯N, and yi applying a onehot method to each encoding, each y is one of 7 labels. After studying weighted loss functions such as sample balance and category balance, it is finally decided to use the multi-classification cross-entropy function in Equation 4–3 as the loss function, and use the compute\_class\_weight function of the sklearn library to calculate the proportion of the loss of different categories of the data set Weight, get loss\_weight ={’ age\_degeneration ’: 1.3, ’ cataract’: 1.2, ’diabetes’: 0.9, ’glaucoma’: 1.2, ’ hypertension’: 1.6, ’myopia’: 1.2, ’normal’: 0.5} , in During model training, set the class\_weight of model.fit to loss\_weight to achieve the effect of weighting various losses.

Visualize the features extracted by the two branches of CNN-Trans respectively. (1) CNN-LSTM branch. Visual analysis of neural networks is of great importance in both research and practical applications. For the model developed in research, not only can automatic classification be achieved, but also the principles behind its behavior should be familiarized so that clinicians can trust and use it without worry. Visualizations can more accurately distinguish classes, better reveal the trustworthiness of classifiers, and help identify bias in datasets.

In recent years, Vision Transformer networks have become the main tool for traditional computer vision tasks, such as object detection [33] and image recognition[34,35]. The importance of the Vision Transformer network creates an urgent need to visualize its decision-making process. This visualization can help debug models, help verify that models are fair and unbiased, and enable downstream tasks. However, there is still very little literature on Transformer visualization for image classification, and several factors prevent its use in visualization applications developed for other forms of neural networks compared to CNNs: this includes inactive activation functions. use, frequent use of skip connections, and the challenge of modeling multiplication used in self-attention. the Vision Transformer network is the self-attention layer [36,37], which assigns a pairwise attention value between every two tokens. In NLP, a token is usually a word or part of a word. Visually, each token can be associated with a patch, and “self-attention” combines information from participating embeddings into the focal embedding representation of the next layer. Therefore, information from different tokens is increasingly mixed in various layers of Vision Transformer. In order to visualize the part of the image that leads to some classification, a common practice is to rely on the attention map obtained by visualizing the Vision Transformer. This paper draws the relevant attention map through the visualize.attention\_map function of the vitkeras module. For the CNN-Trans model, Fig. 13 shows the four-category results of some sample images in the test set in the dataset D2. Three images are randomly selected for each category, a total of 12 images, where the font is green to indicate the predicted label of the model Consistent with the real label is correct, and the font is red to indicate that the predicted label of the model is inconsistent with the real label, which is an error. Fig. 14 presents some qualitative results of ViT attention-weighted activation maps. We observed that the lesion area judged by the model was basically consistent with that judged by clinicians. For example, the clinical diagnosis of cataract mainly depends on whether the fundus image is overall blurred, and the light and dark areas of the attention map are relatively average, while the diagnosis of glaucoma mainly focuses on the optic disc area. By visualizing the features extracted by the two branches of CNNTrans, we found that: CNN-Trans can not only extract representative local features by the coordinated attention mechanism of CNN-LSTM branch, but also by the self-attention of ViT branch. The mechanism extracts global features complementary to local information, and finally, local features are concatenated with global features and fused to generate a more comprehensive feature representation, which in turn improves the final classification accuracy.

**References**

[1] Yu. Lun, W. Lifang, P. Lin, Research progress of fundus image registration technology, J. Biomed. Eng. 28 (05) (2011) 1043–1047.

[2] Signal processing and machine learning for biomedical big data[M]. CRC press, 2018.

[3] Sengupta S, Singh A, Leopold H A, et al. Application of Deep Learning in Fundus Image Processing for Ophthalmic Diagnosis–A Review. arXiv preprint arXiv: 1812.07101, 2018.

[4] D. Socia, C.J. Brady, S.K. West, et al., Detection of trachoma using machine learning approaches, PLoS Negl. Trop. Dis. 16 (12) (2022) e0010943.

[5] P.J. Foster, Y. Jiang, Epidemiology of myopia, Eye 28 (2) (2014) 202–208.

[6] S. Faizal, C.A. Rajput, R. Tripathi, et al., Automated cataract disease detection on anterior segment eye images using adaptive thresholding and fine tuned inceptionv3 model, Biomed. Signal Process. Control 82 (2023) 104550.

[7] I. Qureshi, J. Ma, Q. Abbas, Recent development on detection methods for the diagnosis of diabetic retinopathy, Symmetry 11 (6) (2019) 749.

[8] Chea N, Nam Y. Classification of fundus images based on deep learning for detecting eye diseases. 2021.

[9] J. Wen, D. Liu, Q. Wu, et al., Retinal image-based artificial intelligence in detecting and predicting kidney diseases: Current advances and future perspectives, View (2023) 20220070.

[10] J.Y. Choi, T.K. Yoo, J.G. Seo, et al., Multi-categorical deep learning neural network to classify retinal images: A pilot study employing small database, PLoS One 12 (11) (2017) e0187336.

[11] A. Diaz-Pinto, S. Morales, V. Naranjo, et al., CNNs for automatic glaucoma assessment using fundus images: an extensive validation, Biomed. Eng. Online 18 (2019) 1–19.

[12] J.J. Balaji, A. Agarwal, R. Raman, et al., Comparison of foveal avascular zone in diabetic retinopathy, high myopia, and normal fundus images[C]//Ophthalmic Technologies XXX, SPIE 11218 (2020) 86–97.

[13] M.S. Junayed, M.B. Islam, A. Sadeghzadeh, et al., CataractNet: An automated cataract detection system using deep learning for fundus images, IEEE Access 9 (2021) 128799–128808.

[14] M.M. Butt, D.N.F.A. Iskandar, S.E. Abdelhamid, et al., Diabetic Retinopathy Detection from Fundus Images of the Eye Using Hybrid Deep Learning Features, Diagnostics 12 (7) (2022) 1607.

[15] Shyamalee T, Meedeniya D. CNN based fundus images classification for glaucoma identification[C]//2022 2nd International Conference on Advanced Research in Computing (ICARC). IEEE, 2022: 200-205.

[16] A.E. Ilesanmi, T. Ilesanmi, A.G. Gbotoso, A systematic review of retinal fundus image segmentation and classification methods using convolutional neural networks, Healthcare Analytics 100261 (2023).

[17] Z. Lu, J. Miao, J. Dong, et al., Automatic Multilabel Classification of Multiple Fundus Diseases Based on Convolutional Neural Network With Squeeze-andExcitation Attention, Transl. Vis. Sci. Technol. 12 (1) (2023) 22.

[18] W. Wang, S. Liu, H. Xu, et al., COVIDX-LwNet: A Lightweight Network Ensemble Model for the Detection of COVID-19 Based on Chest X-ray Images, Sensors 22 (21) (2022) 8578.

[19] E. Tartaglione, C.A. Barbano, C. Berzovini, et al., Unveiling covid-19 from chest xray with deep learning: a hurdles race with small data, Int. J. Environ. Res. Public Health 17 (18) (2020) 6933.

[20] C.F. Xception, Deep Learning with Depthwise Separable Convolutions[c]// proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017:) 1251–1258.

[21] K. Greff, R.K. Srivastava, J. Koutník, et al., LSTM: A search space odyssey, IEEE Trans. Neural Networks Learn. Syst. 28 (10) (2016) 2222–2232.

[22] Hou Q, Zhou D, Feng J. Coordinate attention for efficient mobile network design. arXiv 2021. arXiv preprint arXiv:2103.02907, 2021.

[23] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[24] Carion N, Massa F, Synnaeve G, et al. End-to-end object detection with transformers[C]//Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. Springer International Publishing, 2020: 213-229.

[25] Joshi P, Masilamani V. An Efficient Transfer Learning Based Approach for Detecting the Abnormal Fundus Images[C]//2021 5th Conference on Information and Communication Technology (CICT). IEEE, 2021: 1-5.

[26] Raza A, Khan M U, Saeed Z, et al. Classification of eye diseases and detection of cataract using digital fundus imaging (DFI) and inception-V4 deep learning model [C]//2021 International Conference on Frontiers of Information Technology (FIT). IEEE, 2021: 137-142.

[27] C.J. Lai, P.F. Pai, M. Marvin, et al., The Use of Convolutional Neural Networks and Digital Camera Images in Cataract Detection, Electronics 11 (6) (2022) 887.

[28] A. Smitha, P. Jidesh, Classification of multiple retinal disorders from enhanced fundus images using semi-supervised GAN, SN Computer Science 3 (2022) 1–11.

[29] Y. Pan, J. Liu, Y. Cai, et al., Fundus image classification using Inception V3 and ResNet-50 for the early diagnostics of fundus diseases, Front. Physiol. 14 (2023) 160.

[30] A. Shamsan, E.M. Senan, H.S.A. Shatnawi, Automatic Classification of Colour Fundus Images for Prediction Eye Disease Types Based on Hybrid Features, Diagnostics 13 (10) (2023) 1706.

[31] M.A. Ali, M.S. Hossain, M.K. Hossain, et al., AMDNet23: Hybrid CNN-LSTM Deep Learning Approach with Enhanced Preprocessing for Age-Related Macular Degeneration (AMD) Detection, Intelligent Systems with Applications 200334 (2024).

[32] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy 22–29 (October 2017) 618–626.

[33] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. Endto-end object detection with transformers. arXiv preprint arXiv:2005.12872, 2020.

[34] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Proceedings of the 37th International Conference on Machine Learning, volume 1, 2020.

[35] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[36] A. Parikh, Oscar Tackstr ¨om, Dipanjan Das, and Jakob ¨Uszkoreit. A decomposable attention model for natural language inference, in: In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 2249–2255.

[37] J. Cheng, L.i. Dong, M. Lapata, Long shortterm memory-networks for machine reading, in: In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 551–561.

[38] Y. Gal, Z. Ghahramani, Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning 20–22 (2016) 1050–1059.