NATIONAL ECONOMICS UNIVERSITY
FACULTY OF MATHEMATICAL ECONOMICS

MAJOR ASSIGNMENT

Course: Deep Learning

TOPIC: HISTOPATHOLOGIC CANCER DETECTION

Instructor: **Dr. Nguyen Thi Kim Ngan**

Students:	**Lam Minh Quang**	**11225430**
	Phan Dai Tung	**11226767**
	Nguyen Minh Hieu	**11222333**
	Than Quang Vinh	**11226938**

Hanoi, October 2025

Table of Contents

PART I. THEORY

Convolutional Neural Networks (CNNs)

1. Motivation of CNNs

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision, particularly in tasks like image recognition and object detection. Their design is specifically motivated by the inherent properties of image data and the limitations of traditional neural network architectures when applied to such data.

Traditional Multi-Layer Perceptrons (MLPs), or fully connected neural networks, are poorly suited for image data because they treat inputs as flat vectors, ignoring the two-dimensional structure of images. Flattening an image into a vector leads to extremely high dimensionality, producing millions of parameters that are computationally expensive to train and prone to overfitting. Moreover, this process discards spatial relationships between pixels, preventing the model from exploiting local patterns such as edges or textures. Finally, MLPs lack translation invariance: an object recognized in one location must be relearned if it appears elsewhere in the image. These limitations highlight the inefficiency of traditional neural networks for vision tasks and motivate the development of architectures like Convolutional Neural Networks, which explicitly incorporate spatial structure, parameter sharing, and translation equivariance.

More formally, Convolutional Neural Networks (CNNs) were introduced to overcome the inefficiencies of traditional fully connected architectures in processing visual data. Unlike Multi-Layer Perceptrons, CNNs exploit the two-dimensional spatial structure of images through three core principles: (i) sparse interactions, (ii) parameter sharing, and (iii) equivariance to translation.

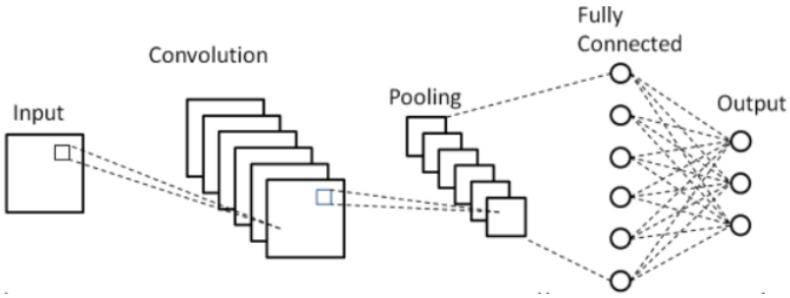
- **Sparse interactions** are achieved through kernels (filters) that are significantly smaller than the entire image, ensuring each output element depends solely on a localized receptive field. By constraining the connections per output to a value kkk rather than the full input size mmm (where $k \ll m$), the parameter count and computational complexity are diminished from $O(m \times n)$ to $O(k \times n)$, yielding improvements in both runtime performance and statistical efficiency by focusing learning on relevant local patterns.
- **Parameter sharing** implies that the identical kernel is applied across various spatial locations in the image; this not only conserves memory but also facilitates the detection of the same feature (e.g., an edge in a specific orientation)

irrespective of its position. Consequently, convolutional layers produce feature maps where learned parameters exhibit spatial repetition, aligning well with the recurrent motifs found in natural images. This mechanism further allows gradients to accumulate from diverse locations during backpropagation, rendering the optimization of filters more effective compared to training isolated weights in an FCNN.

- The convolution operation inherently possesses **equivariance** with respect to translations: if an object in the input image is shifted by a certain amount, the corresponding feature map shifts identically. This property is vital for many visual tasks, as the emphasis is on detecting a feature's presence rather than its absolute position. To achieve invariance to smaller translations, networks frequently integrate pooling operations (e.g., max-pooling), which summarize regional statistics to diminish sensitivity to minor perturbations, thereby augmenting the model's generalization capabilities. Nonetheless, equivariance and pooling constitute strong inductive biases: they confer advantages when the assumptions about translations hold true, but may induce underfitting if the problem necessitates preserving precise locations or invariance to alternative transformations, such as scaling or rotation, which standard convolutions do not inherently support.

The combination of these principles established CNNs as the foundation of modern computer vision and made them particularly well-suited for domains such as medical imaging, where data is high-dimensional yet often limited in quantity.

2. Architecture of CNNs



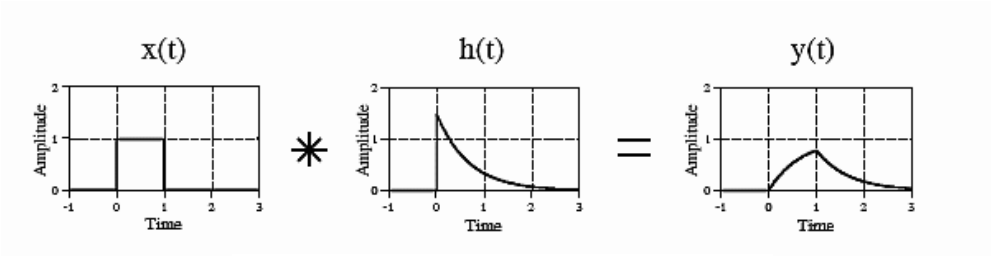
The architecture of Convolutional Neural Networks (CNNs) is specifically designed to address the challenges of modeling high-dimensional, grid-structured data such as images. CNNs exploit the spatial locality of image data by restricting connections to small receptive fields and leveraging parameter sharing through convolutional kernels. This design drastically reduces the number of trainable parameters, enhances statistical efficiency, and preserves spatial hierarchies of features across layers.

A typical CNN is composed of three primary types of layers—**convolutional layers**, **pooling layers**, and **fully connected layers**—which, when stacked sequentially, form deep architectures capable of extracting increasingly abstract representations of input data.

2.1. Convolutional layers

The Convolution Operation

In mathematics, convolution is a mathematical operation on two functions f and g that produces a third function $f * g$, as the integral of the product of the two functions after one is reflected about the y-axis and shifted. The term convolution refers to both the resulting function and to the process of computing it. The integral is evaluated for all values of shift, producing the convolution function.



The convolution of f and g is written $f * g$, denoting the operator with the symbol $*$. It is defined as the integral of the product of the two functions after one is reflected about the y-axis and shifted. As such, it is a particular kind of integral transform:

$$(f * g)(t) := \int f(\tau)g(t-\tau)d\tau$$

Convolutional layers

In the context of images, convolution is performed between an input image (a two-dimensional grid of pixel values) and a small matrix of weights called a kernel or filter. This operation generalizes to:

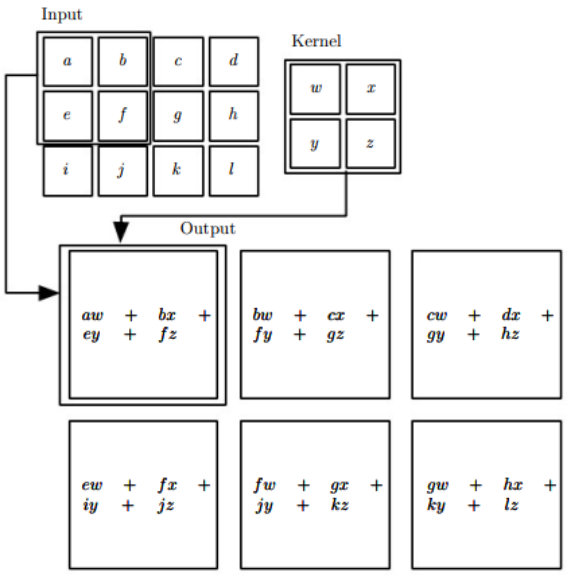
$$S(i,j) = (I,K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

where I denotes the input image represented as a matrix
K is the convolution kernel (or filter)
$S(i,j)$ is the output feature map value at the spatial location (i,j)
The indices m and n iterate over the spatial extent of the kernel

This formula defines how a convolutional kernel interacts with an image to produce a new representation that highlights certain patterns. At each output position (i,j),

the kernel K is overlaid on a local neighborhood of the image I. Each kernel element $K(m,n)$ is multiplied by the corresponding pixel $I(i+m, j+n)$, and the results of these element-wise multiplications are summed together. This weighted sum produces a single scalar value $S(i,j)$, which encodes the presence and strength of the pattern represented by the kernel in that local region. The process is repeated systematically as the kernel slides across the image, generating the complete feature map S. Intuitively, the kernel acts as a pattern detector: for example, an edge-detection kernel amplifies intensity differences along certain orientations, while a smoothing kernel reduces local variations. In deep learning, the kernel weights are not fixed but learned, allowing the network to automatically discover and extract features that are most useful for the task at hand.

The matrix-based perspective clearly shows that convolution is essentially an element-wise multiplication between the kernel and a local image patch, followed



by summing all the products. This process converts local pixel neighborhoods into single values, which together make up the feature map.

Convolutional layers are also able to significantly reduce the complexity of the model through the optimization of its output. These are optimised through three hyperparameters: the **depth**, the **stride,** and the setting of **zero-padding**

- The **depth** of a convolutional layer's output volume is determined by the number of filters (or kernels) applied to the input. Each filter is designed to capture a distinct feature, such as edges, textures, or color patterns, producing a unique

activation map. When these activation maps are stacked along the depth dimension, they collectively form the output volume. The depth, therefore, represents the number of distinct feature detectors learned at a given layer. This characteristic is crucial for effective feature extraction and hierarchical learning: individual filters respond to specific local patterns, while the aggregation of multiple filters enables the network to construct increasingly abstract and complex representations of the input.

- The **stride** is a fundamental hyperparameter that determines the step size of the filter (or kernel) as it traverses the input image during the convolution operation. Specifically, it defines how many pixels the filter shifts horizontally and vertically at each step. The stride directly influences the spatial dimensions of the resulting feature map, the computational cost of the operation, and the granularity of feature extraction.
- **Padding** is the technique of adding extra pixels—usually zeros—around the border of an input image or feature map to manage spatial dimensions and preserve important information during convolution. Without padding, each convolutional operation reduces the size of the feature map, and in deep architectures, this progressive shrinking could eventually collapse the data to a single pixel, hindering the network's ability to learn effectively. Padding preserves spatial resolution across layers, ensuring that feature extraction remains consistent even at image boundaries. Moreover, it ensures that edge and corner pixels receive equal attention during convolution, allowing filters to capture features near the image boundaries as effectively as those in the center. By doing so, padding not only stabilizes spatial resolution but also preserves valuable edge information critical for accurate feature extraction.

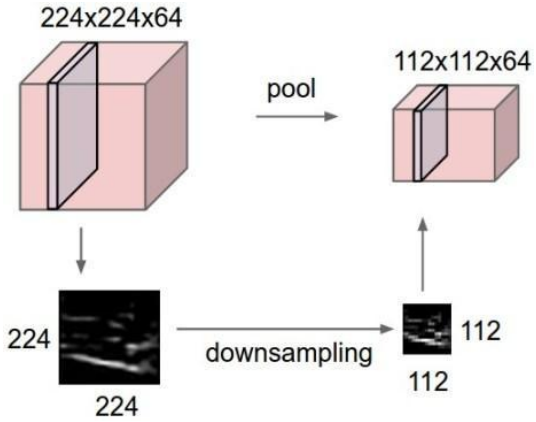
It is important to understand that through using these techniques, we will alter the spatial dimensionality of the convolutional layers output. To calculate this, we can make use of the following formula:

$$O = \frac{V - R + 2Z}{S} + 1$$

Where V represents the input volume size (height×width×depth), R represents the receptive field size, Z is the amount of zero padding set, and S refers to the stride.

2.2. Pooling layers

Pooling is a downsampling operation applied to feature maps to reduce their spatial dimensions while retaining the most important information. It involves sliding a small two-dimensional filter over each channel of the feature map and aggregating the values within the covered region, typically using either the maximum or the average. **Max pooling** selects the highest value in each region, preserving the most prominent features and enhancing sharpness, while **average pooling** computes the mean of the region, resulting in smoother representations that retain the overall essence of the features.



Pooling serves several critical functions in CNNs. It **reduces the spatial dimensions** of feature maps, thereby decreasing the number of parameters and computational cost. This dimensionality reduction also helps **control overfitting** by limiting model complexity. Moreover, pooling contributes to **translational invariance**, enabling the network to recognize features regardless of minor shifts or distortions in their spatial position. By consolidating local features into a more abstract representation, pooling layers enhance the network's efficiency, stability, and robustness in extracting dominant, invariant patterns from visual data.

2.3. Fully connected layers

A **fully connected layer** is a layer in which every neuron is connected to all neurons in the preceding layer. This dense connectivity allows the network to learn global relationships among features, enabling it to integrate and interpret the information extracted by earlier convolutional and pooling layers. Unlike convolutional layers that focus on local spatial patterns, fully connected layers combine all learned features to form high-level, abstract representations of the input data.

The operation of a fully connected layer involves multiplying each input by a corresponding weight, adding a bias, and passing the result through a non-linear activation function to generate the output. This process allows the layer to model complex feature interactions and learn discriminative patterns useful for decision-making. In image classification tasks, for instance, the outputs from the convolutional and pooling layers are flattened into a single vector and fed into one or more fully connected layers, which ultimately perform the classification or regression. Thus, the fully connected layer acts as the final stage of feature synthesis and prediction in a CNN, translating learned spatial features into actionable outputs.

PART II. PRACTICAL APPLICATION

Problem Description

Histopathologic Cancer Detection – Automated Identification of Metastatic Breast Cancer in Lymph Node Tissue Slides

Computer Vision (CV) and Deep Learning have gone beyond entertainment and commercial applications to become essential tools for addressing large-scale social issues. Fields such as Healthcare, Food Security, and Occupational Safety are increasingly integrating CV to automate processes, enhance accuracy, and minimize human risk. Among these domains, Healthcare is often given top priority due to its direct impact on human life and quality of life, demanding an urgent need for near-perfect accuracy.

To meet the requirement for a project that is both practical, engaging, and socially significant, this report focuses on the field of clinical diagnosis support—where human errors can lead to serious consequences. The selected topic is **Histopathologic Cancer Detection**, specifically the identification of metastatic cancer tissue in histopathological images.

1.1. Research Objectives

The main objective of this research is to develop a computer vision system using deep learning to automatically detect metastatic cancer cells in histopathologic images of lymph node tissues. Specific goals include:

- Enhancing diagnostic accuracy by supporting pathologists with an automated detection tool.
- Reducing human error and inter-observer variability in clinical diagnosis.

- Demonstrating the social and medical impact of artificial intelligence (AI) in life-critical healthcare applications.
- Implementing and evaluating a model that achieves near-clinical performance using benchmark datasets from Kaggle and international medical research.

1.2. Medical Background and Social Necessity

Pathological Diagnosis of Metastatic Breast Cancer

In clinical medicine, determining the stage of breast cancer is a crucial step that often involves biopsy and examination of the sentinel lymph nodes. The presence of metastatic cancer cells, even the smallest micrometastases, is a decisive factor in defining the patient's subsequent treatment plan — including radiotherapy, chemotherapy, or additional surgical procedures.

Pathologists face a significant challenge: the enormous workload and the pressure of examining extremely high-resolution images known as **Whole-Slide Imaging (WSI).** A single WSI can contain billions of pixels. Manually inspecting each region of the slide to identify faint or small metastatic areas can lead to fatigue and the risk of missed diagnoses, especially in early-stage metastasis cases.

Social Impact of Automated Diagnosis

Machine vision systems are designed to serve as **Computer-Aided Diagnosis (CAD)** tools — acting as an objective, tireless "second checker." The potential of such systems to improve healthcare quality has been clearly demonstrated.

A key study showed that when deep learning system predictions were combined with pathologists' diagnoses, classification performance improved significantly. Specifically, follow the research of Deep Learning for Identifying Metastatic Breast Cancer, the **Area Under the ROC Curve (AUC)** increased to **0.995**. Most importantly, AI intervention reduced human error in identifying metastatic regions by approximately **85%**.

This **85% error reduction** is not merely a technical achievement; it is evidence of AI's capacity to enhance the quality of clinical diagnosis. In a medical environment that demands the highest levels of reliability and precision, minimizing the risk of missed diagnoses represents an invaluable social contribution.

The CAD system helps reduce **inter-observer variability**, standardize diagnostic procedures, and ensure that patients receive accurate and timely treatment plans. Consequently, **computer vision–based cancer detection** remains a central focus

of ongoing research, with continuous advancements observed from 2016 to the latest deep learning techniques in 2024.

1.3. Input of the Problem

The input consists of high-resolution histopathologic images (Whole Slide Images – WSIs) of lymph node tissue samples. Each image tile (patch) is labeled as either:

- **Positive**: containing metastatic tumor cells
- **Negative**: normal or non-metastatic tissue

The dataset is publicly available from **the Kaggle Histopathologic Cancer Detection Challenge**, based on medical imaging data standardized by international pathology institutions.

1.4. Output of the Problem

The expected output of the system is:

- The classification threshold was automatically optimized to achieve high TPR, ensuring the model prioritizes correctly identifying positive cases.
- The model achieved strong overall performance, with key metrics such as Accuracy, Precision, Recall, F1-score, and ROC AUC reported in detail for each class.
- Confusion Matrix and ROC Curve visualizations were generated to illustrate the model's performance and discriminative ability.
- A comprehensive evaluation summary was produced, including all quantitative metrics and the optimal threshold, supporting model comparison and final report analysis.

1.5. Summary of the Tasks Performed for this Problem

a. Problem Identification and Motivation:

Recognized the growing need for automated cancer detection systems to reduce diagnostic errors and workload in clinical pathology, especially in breast cancer staging.

b. Literature and Medical Review:

Studied the diagnostic process of breast cancer metastasis in lymph nodes and the challenges faced by pathologists in manually reviewing large-scale WSIs.

c. Dataset Exploration and Preprocessing:

- Loaded and visualized the Kaggle histopathologic dataset.
- Performed image normalization, resizing, and augmentation.
- Split data into training, validation, and test sets.

d. Model Development:

Compare 3 methods to evaluate their performance in image classification tasks. The tested methods include:

- A baseline Convolutional Neural Network (CNN)
- A CNN enhanced with transfer learning using pretrained weights, models incorporating data augmentation and fine-tuning strategies
- The Vision Transformer (ViT) architecture.

The main goal of these experiments is to determine whether traditional CNN architectures can still compete with newer transformer-based models under the same conditions.

e. Model Evaluation:

- The classification threshold was automatically optimized to achieve high TPR, ensuring the model prioritizes correctly identifying positive cases.
- The model achieved strong overall performance, with key metrics such as Accuracy, Precision, Recall, F1-score, and ROC AUC reported in detail for each class.
- Confusion Matrix and ROC Curve visualizations were generated to illustrate the model's performance and discriminative ability.
- A comprehensive evaluation summary was produced, including all quantitative metrics and the optimal threshold, supporting model comparison and final report analysis.

f. Social and Clinical Significance:

- Highlighted the system's potential as a Computer-Aided Diagnosis (CAD) tool to improve reliability, standardization, and timeliness in cancer diagnosis.

- Emphasized its contribution to reducing human fatigue, minimizing diagnostic risk, and ultimately improving patient outcomes.

Dataset Description

Histopathologic Cancer Detection: https://www.kaggle.com/competitions/histopathologic-cancer-detection/data

1. **Dataset introduction**
   The Histopathologic Cancer Detection dataset from Kaggle is used for a binary classification task of H&E-stained histopathology images. Each sample is a 96×96 pixel (RGB) patch cropped from a larger histology image, accompanied by a binary label: 1 indicates the presence of cancer cells, and 0 indicates their absence. The H&E staining characteristics and imaging conditions cause significant variations in color, brightness, and contrast between images, which need to be addressed during preprocessing and data augmentation.

2. **Data Structure and Format**
   The training set contains 220,025 labeled images, while the test set contains 57,458 images. The label distribution in the training set is imbalanced: 89,117 positive images (approximately 40.5%) and 130,908 negative images (approximately 59.5%). Due to H&E staining, the data exhibits optical variability in hue, brightness, and contrast. Additionally, some patches may be nearly monochromatic (very light or dark), blurred, or contain artifacts.

   The dataset consists of three main components:
   - The train/ folder contains images in TIFF format, with filenames in the format id.tif and expected dimensions of 96×96×3.
   - The train_labels.csv file has two columns, id and label, mapping each image in train/ to a binary label (0 or 1).
   - The test/ folder contains unlabeled images for evaluation purposes.

3. **Dataset split**
   Due to the imbalanced dataset, we decided to sample 89,000 images for each label, resulting in a total of 178,000 images. Including:
   - Training data: 128,160 images
   - Validation data: 32,040 images
   - Test data: 17,800 images (provided without labels)

CNN Model Design

1. CNN Base model

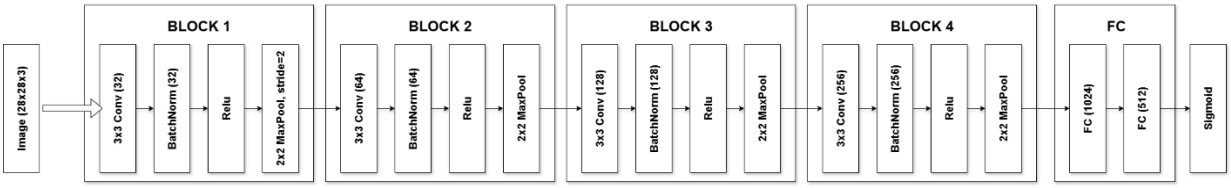
Model Architecture

We design a Convolutional Neural Network (CNN) tailored for binary histopathology image classification. The feature extractor consists of five sequential convolutional blocks (indexed 1–5), where the i-th block applies a 2D convolution with C_i output channels ($C = [32, 64, 128, 256, 512]$) and kernel size k_i (typically 3×3 except one 2×2 kernel in block 2). Each convolution is immediately followed by batch normalization and a ReLU nonlinearity, and then a 2×2 max-pooling layer to downsample by a factor of 2. Concretely, for the block k, if X_{k-1} is the input tensor of shape $(C_{k-1}, H_{k-1}, W_{k-1})$, we compute

$$X_k = MaxPool\left(ReLU\big(BatchNorm(W_k * X_{k-1} + b_k)\big)\right)$$

where $*$ denotes convolution. For example, the first block uses a 3×3 conv layer mapping 3 input channels to 32 filters, and the fifth block uses a 3×3 conv layer mapping 256 channels to 512 filters. After five blocks, the spatial dimensions are reduced (for 96×96 inputs, eventually to 3×3) and the feature depth is 512, yielding a flattened feature vector of length $512 \times 3 \times 3 = 4608$.

The final classifier consists of three fully connected (FC) layers. The flattened convolutional features are first fed to an FC layer of size 1024 with ReLU activation and dropout (p=0.4), then to a second FC of size 512 with ReLU and dropout (p=0.4), and finally to a single output unit with a Sigmoid activation. Thus, the network outputs a probability $\hat{y} \in (0,1)$ of the "cancer" class. In summary, the model pipeline is:



This deep stack of convolutional filters and non-linearities allows the model to learn hierarchical tissue features (from edges to complex patterns) while gradually reducing spatial resolution. Batch normalization stabilizes training, and dropout mitigates overfitting in the FC layers. The Sigmoid output yields a probability $\hat{y} = \sigma(z)$, where $\sigma(x) = 1/(1 + e^{-x})$, for the binary classification task.

Training Procedure

Training is performed for up to 30 epochs. We employ the AdamW optimizer with a learning rate of 1.5×10^{-4} and weight decay of 10^{-4}. The loss function is binary cross-entropy (BCE) between the predicted probability $\hat{y}_i$ and true label $y_i \in (0,1)$:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \; log \; log \; (\hat{y}_i) + (1 - y_i) \; log \; log \; (1 - \hat{y}_i) \;]$$

averaged over the mini-batch (BCE as in **nn.BCELoss**). Model weights are updated via backpropagation of this loss, using AdamW's adaptive moment estimates and weight decay regularization (in effect, decoupled L2 penalty) at each step.

During training, we monitor both the loss and the Area Under the ROC Curve (AUC) on the training and validation sets. The AUC is a threshold-independent metric assessing the trade-off between true positive rate and false positive rate. Specifically, AUC is the area under the Receiver Operating Characteristic curve (integral of TPR vs. FPR). We record AUC and loss each epoch, and save the model state with the lowest validation loss.

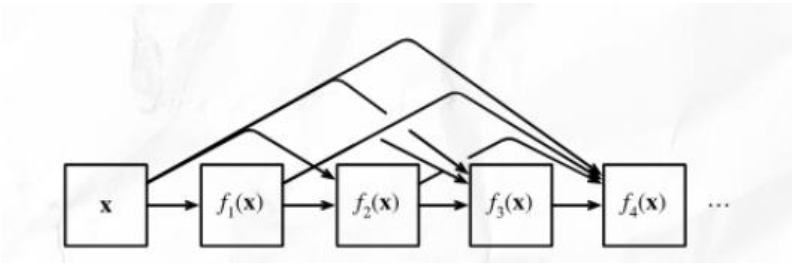
2. Fine-tuning Model DensNet-169

Model Architecture

We adopt **DenseNet-169**, a deep convolutional neural network originally proposed by Huang et al. (2017), as the backbone for binary histopathology image classification.

DenseNet (Densely Connected Convolutional Network) introduces dense connectivity among layers: each layer receives the feature maps of all preceding layers as input, and its own output is passed to all subsequent layers. This structure encourages feature reuse, strengthens gradient flow, and reduces the number of parameters compared to traditional CNNs of similar depth.

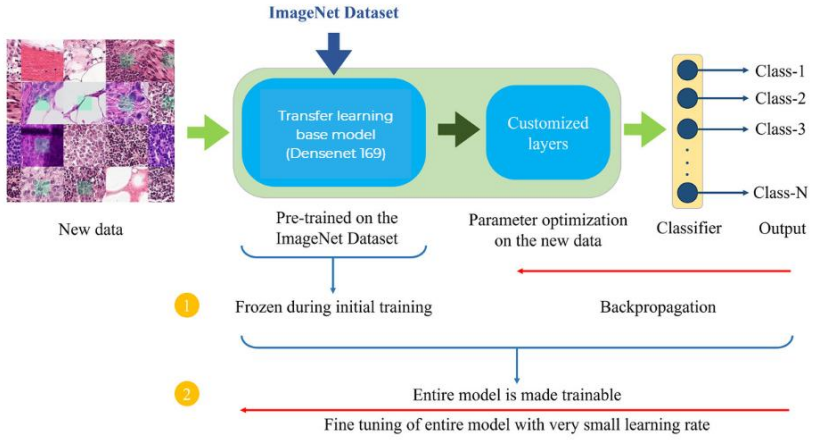
Formally, for a dense block consisting of L layers, the l-th layer receives as input the concatenation of all preceding feature maps:



$$x_l = H_l([x_0, x_1, \ldots, x_{\{l-1\}}])$$

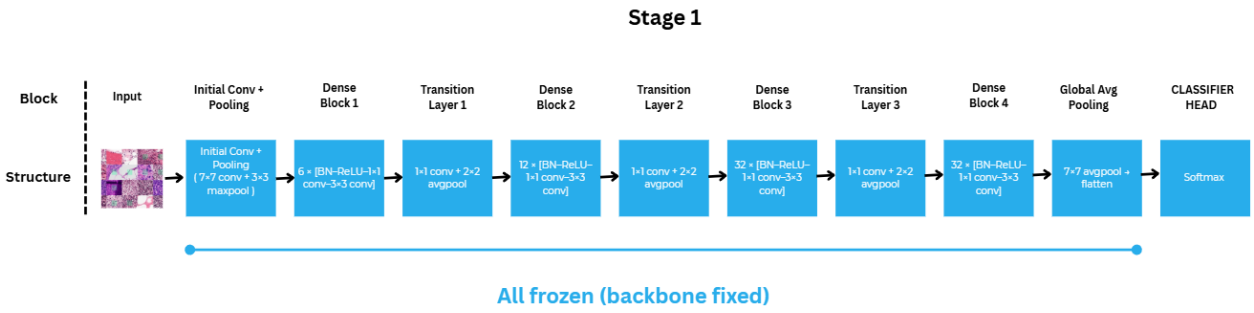
Where $H_l(.)$ denotes the composite function of batch normalization, ReLU, and convolution operations.

DenseNet-169 [1 + (6+12+32+32)×2 + 3 + 1 = 169] contains four dense blocks interleaved with transition layers (each performing 1×1 convolution and 2×2 average pooling). The architecture begins with an initial 7×7 convolution followed by max-pooling and ends with a global average pooling layer and a fully connected classification head.



For transfer learning, we initialize DenseNet-169 with ImageNet pre-trained weights, preserving its robust low-level and mid-level feature representations. The final fully connected layer (originally with 1000 outputs) is replaced by a single output neuron with a Sigmoid activation for binary classification. Thus, the model outputs a probability p ∈ [0,1] representing the likelihood of the positive ("cancer") class.

Training Procedure (Stage 1)



For initial training, we use the Fastai cnn_learner (formerly create_cnn) with the DenseNet-169 backbone and a dropout rate of 0.5 in the final classification layer. The AdamW

optimizer is employed, which decouples weight decay from gradient updates, providing better generalization. The primary loss function is Binary Cross-Entropy.

We conduct a learning rate finder across multiple weight decay (WD) settings (1e-6, 1e-4, and 1e-2) for 600 iterations each. This process identifies the optimal learning rate that minimizes loss while maintaining training stability. Visualization of loss vs. learning rate (in log-scale) allows us to select max_lr = 2e-2 and wd = 1e-4 as the optimal hyperparameters.

1-Cycle Policy

Training follows the 1-cycle policy over 8 epochs, where the learning rate is gradually increased and then decreased. This cyclic schedule encourages faster convergence and reduces overfitting.

The learning rate $\eta(t)$ at iteration t is adjusted using a cosine annealing schedule, which smoothly increases and then decreases the learning rate within one cycle. The mathematical formulation is expressed as:

$$\eta(t) = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left[1 + cos\ cos\ \left(\frac{\pi t}{T}\right)\right]$$

where:

- η_{max} and η_{min} denote the minimum and maximum learning rates,
- T is the total number of iterations in the cycle.

During the **warm-up phase** (t ∈ [0,T/2]), the learning rate increases linearly or semi-cosinely from η_{max} to η_{min}. In the **annealing phase** (t ∈ [T/2, T]), it symmetrically decreases back to a low value.

This approach enables faster initial learning while ensuring stable convergence toward the end of training.

Momentum Schedule

Momentum, denoted by m(t) is adjusted inversely to the learning rate. When the learning rate increases, momentum decreases to stabilize gradient updates; conversely, when the learning rate decreases, momentum increases to accelerate convergence. The schedule can be modeled as:

$$m(t) = m_{max} - \frac{1}{2}(m_{max} - m_{min})\left[1 + cos\ cos\ \left(\frac{\pi t}{T}\right)\right]$$

where:

- m_{max} and m_{min}represent the upper and lower bounds of momentum.

This inverse scheduling of momentum complements the learning rate dynamics, resulting in smoother and more stable training.

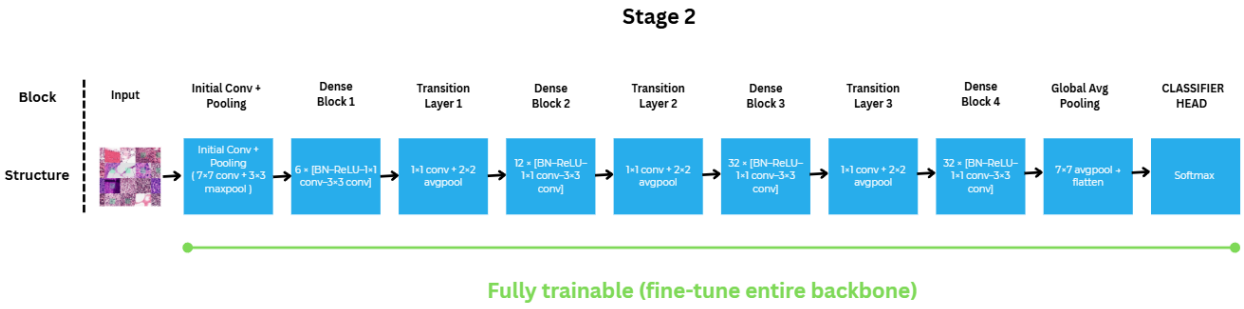
Weight Decay Schedule

An optional extension of the 1-Cycle Policy involves modulating **weight decay** $\lambda(t)$concurrently with learning rate changes. Typically, the weight decay coefficient decreases slightly during the high learning-rate phase (to allow flexibility) and increases again when the learning rate is reduced (to reinforce regularization). The simplified functional form is:

$$\lambda(t) = \lambda_{min} + \frac{1}{2}(\lambda_{max} - \lambda_{min})\left[1 - cos\ cos\ \left(\frac{\pi t}{T}\right)\right]$$

This dynamic adjustment provides adaptive regularization that maintains model stability while preventing overfitting.

Fine-Tuning Procedure (Stage 2)



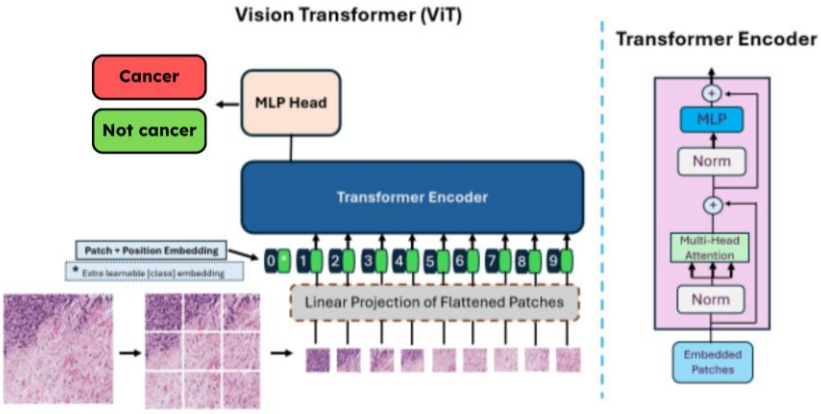
After completing the initial training phase, the model weights obtained from Stage 1 were reloaded to initialize the fine-tuning process. In this stage, the pretrained DenseNet-169 architecture was fully unfrozen, allowing gradient updates across all convolutional layers. This step enabled the network to refine both low-level and mid-level feature representations, thereby adapting more effectively to the target histopathologic image dataset.

To ensure optimal convergence, a new learning rate search was conducted using the Fast.ai learning rate finder, incorporating weight decay regularization to mitigate overfitting. The resulting loss–learning rate curve was analyzed to determine an appropriate discriminative learning rate range for the 1-Cycle training schedule (Smith, 2018). This schedule

dynamically varies the learning rate and momentum throughout the training process to achieve faster convergence and improved generalization.

3. Vision Transformers (ViT_small_patch16_224)

Vision Transformers



We employ a Vision Transformer (ViT) backbone (ViT-Small, patch size 16) pre-trained on ImageNet to extract visual features. In this architecture, each RGB image $I \in R^{H\times W\times 3}$ is first divided into N non-overlapping patches of size $P \times P$ (here $P = 16$), yielding $N = (H/P)(W/P)$ patches. Each patch is flattened to a vector and linearly projected into a d -dimensional embedding space (for ViT-Small, $d = 384$). Formally, for the i -th patch we compute

$$E_i = W\,Flatten(Patch_i) + b, \quad i = 1, \dots, N,$$

where $W \in R^{d\times(PP3)}$ and $b \in R^d$ are learnable parameters. A learnable "classification" token $x_{cls} \in R^d$ is prepended to the sequence of patch embeddings, and positional encodings $p_i \in R^d$ are added to each patch embedding to retain spatial information. The resulting input to the Transformer encoder is

$$z_0 = [\, x_{cls},\ E_1 + p_1,\ \dots,\ E_N + p_N \,].$$

This sequence (length $N + 1$) is passed through L identical Transformer layers, each consisting of multi-head self-attention and feed-forward sublayers (with residual connections and layer normalization).

Within each attention block, the queries Q , keys K , and values V are computed from the input. The output of the self-attention is given by the standard scaled dot-product formula:

$$Attention(Q, K, V) = Softmax\left(\frac{Q\,K^{\top}}{\sqrt{d_k}}\right) V$$

where d_k is the per-head key dimensionality. (Multi-head attention concatenates several such outputs in parallel.) This allows the model to globally aggregate information from all patches. After L layers, we extract the final hidden state z_L^{cls} corresponding to the [CLS]

token. A classification head (an MLP) maps $z_L^{cls} \in R^d$ to a binary output. In our implementation, the original ViT head is replaced by a small feed-forward network (one hidden layer of 256 units with ReLU and dropout $p = 0.3$, followed by a linear output). A sigmoid activation produces the probability $\hat{y}$ of the "cancer" class. This replacement follows the fine-tuning strategy of the original ViT: the pre-trained head is removed and a new head is attached for the downstream task. The model is partially fine-tuned: the patch embeddings and early Transformer blocks are typically frozen (to preserve pretrained features), while later blocks and the new head are trained on histopathology data.

Training Procedure:

The ViT model (initialized from an ImageNet-pretrained checkpoint) is trained for binary classification. We optimize using the AdamW optimizer, which applies decoupled weight decay to improve generalization. The loss function is binary cross-entropy for a batch of B samples. Gradients are computed by backpropagation through the Transformer and head, and parameters are updated iteratively (e.g. learning rate 10^{-4}). We typically freeze the lowest layers and fine-tune the rest to adapt the model to histopathology images. Training proceeds for multiple epochs, monitoring both training and validation loss.

Experimental Results

In this study, two state-of-the-art deep learning architectures — Vision Transformer (ViT) and DenseNet — were trained to classify histopathologic images into two categories: Cancerous (1) and Non-cancerous (0). The primary objective was to maximize the True Positive Rate (TPR), ensuring that the models are highly sensitive to detecting cancerous tissue — a critical requirement in medical diagnosis systems.

To evaluate the model performance in distinguishing between cancerous and non-cancerous tissue, several key metrics were used: Accuracy, Precision, Recall (TPR), F1-score, False Positive Rate (FPR), and Area Under the ROC Curve (AUC-ROC). Furthermore, threshold tuning was applied to prioritize TPR instead of using the default classification threshold of 0.5.

CNN Model Results

The CNN model achieved an **accuracy of 94.13%** and a high **ROC-AUC of 0.9851**, demonstrating strong capability in distinguishing cancerous from non-cancerous tissue. With a **threshold of 0.8591**, it prioritizes sensitivity, achieving a **TPR of 0.95** while keeping the **FPR at 0.0674**.

However, despite its overall reliability, the CNN still shows **a moderate number of false positives**, which may lead to unnecessary follow-up tests in clinical settings. Moreover, compared to transformer-based models like ViT, CNNs may struggle to **capture long-range spatial dependencies** and **subtle texture variations**, limiting performance on more complex histopathologic patterns.

DenseNet Model Results

DenseNet also performed well, with a **TPR of 95.02%**, nearly matching that of the ViT model. However, its **FPR is noticeably higher at 3.27%**, which implies a greater tendency to incorrectly classify non-cancerous samples as cancerous — a drawback that may result in unnecessary follow-up testing or patient stress.

The **Precision** of 96.67% for the cancer class is strong but still lower than ViT. The **AUC-ROC** of 0.9915 shows that DenseNet has solid discriminatory power, though it slightly underperforms compared to ViT.

The classification threshold used (0.5309) is much closer to the default 0.5, indicating that the model is **less conservative** in predicting cancer. While this helps maintain high recall, it also leads to more false positives.

DenseNet proves to be a capable and stable model. However, it lags behind ViT in overall precision and makes more false positive predictions. In settings where false alarms must be minimized, DenseNet may be less ideal compared to more cautious alternatives like ViT.

Vision Transformer (ViT) Model Results

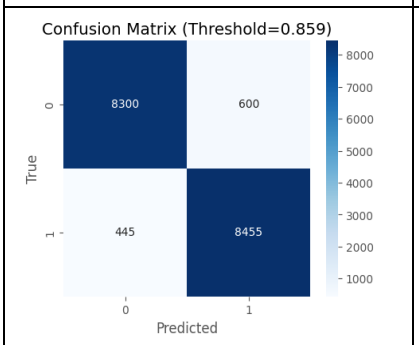
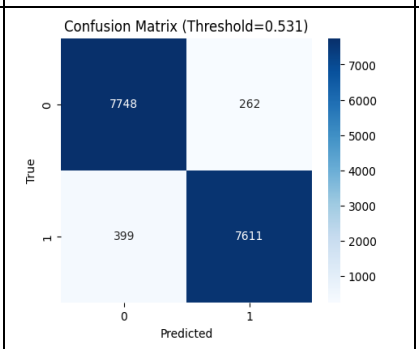
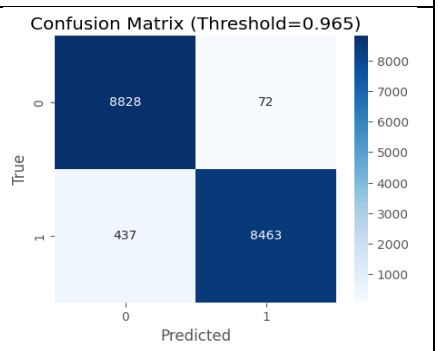
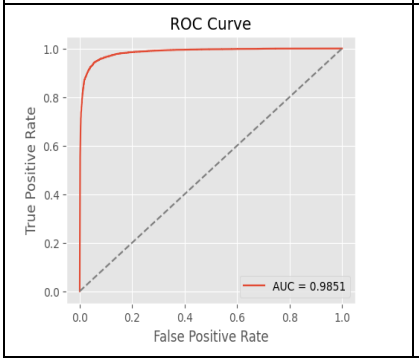
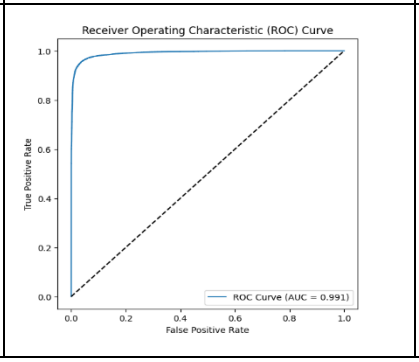
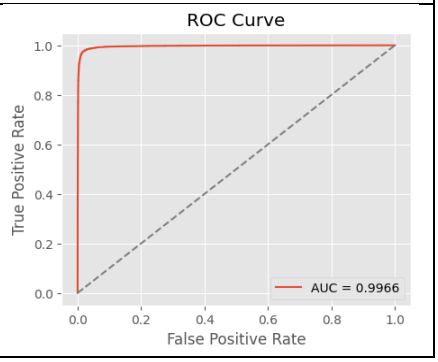
The ViT model achieved outstanding performance with an **AUC-ROC of 0.9966**, indicating an almost perfect ability to distinguish between the two classes. The **precision for the cancerous class is 99.16%**, meaning nearly all predicted cancer cases are truly positive. This is crucial in clinical applications, as it minimizes the risk of false alarms that can lead to unnecessary anxiety and medical procedures.

The **Recall (TPR)** of **95.09%** means the model misses only about 4.91% of actual cancer cases — a relatively low and acceptable rate in many medical screening contexts. Notably, the **FPR is extremely low at just 0.81%**, indicating the model rarely misclassifies healthy tissue as cancerous. This demonstrates the model's **robustness and high reliability**.

The threshold was tuned to **0.9652**, making the model more **conservative and cautious** in labeling a sample as cancerous. This resulted in higher precision and lower FPR but accepted a small increase in false negatives.

Overall, the ViT model exhibits **exceptional performance**, particularly suitable for high-risk applications like cancer diagnosis. Its high overall accuracy of 97.14%, combined with a strong balance between precision and recall, makes it a highly dependable tool for assisting in histopathological cancer detection.

Overall Comparison: ViT vs DenseNet vs CNN

CNN Base model	DenseNet_169	ViT_small_patch16
		
		

Conclusion

This study developed a deep learning-based computer-aided detection (CAD) system for histopathological metastasis detection, focusing on optimizing the true positive rate (TPR) through threshold adjustment using a Kaggle dataset. Three models—CNN, DenseNet-169 with transfer learning, and ViT-Small/16—were evaluated, with emphasis on balancing sensitivity and specificity.

ViT-Small/16 outperformed the other models, achieving an AUC of 0.9966, accuracy of 97.14%, precision of 99.16%, recall of 95.09%, and FPR of 0.81%. DenseNet-169 had a higher FPR (3.27%), and CNN showed the lowest performance with an FPR of 6.74%.

ViT-Small/16 demonstrated the best balance for clinical CAD applications, particularly for screening purposes where clinician oversight is required. The system met the study's objectives, but further validation and refinement are needed for clinical deployment.

References

1. AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE https://arxiv.org/pdf/2010.11929
2. Pooling Methods in Deep Neural Networks, a Review https://arxiv.org/pdf/2009.07485#:~:text=Vectors%20of%20locally%20aggregated%20descriptors,%C3%97%2064%20patches%20%5B26%5D.
3. https://cs231n.github.io/convolutional-networks/
4. [1] Smith, L. N. (2018). *A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay*. arXiv preprint arXiv:1803.09820. Available: https://arxiv.org/abs/1803.09820
5. Mechanism of feature learning in convolutional neural networks https://arxiv.org/pdf/2309.0057