一、Optimization

1. loss function

损失函数是优化过程中最核心的目标函数，其定义为模型预测值与真实值之间的差异。 The loss function is the core objective in optimization, defined as the difference between predicted and true values.

数学形式为： The mathematical form is:

对于某个函数 f(x)，优化的目标是找到最小化它的输入值 x* = argmin f(x)。 For a function f(x), the optimization goal is to find the input value x* = argmin f(x).

在机器学习中，f(x) 就是损失函数或代价函数。 In machine learning, f(x) is the loss function or cost function.

其作用是衡量模型输出与期望输出的偏差，指导模型更新参数。 It measures the deviation between model output and expected output, guiding parameter updates.

2. some common loss functions

常见的损失函数包括： Common loss functions include:

均方误差（Mean Squared Error, MSE）： Mean Squared Error (MSE):

$L = (1/n) \sum (y_i - \hat{y}_i)^2$

交叉熵损失（Cross Entropy Loss）： Cross Entropy Loss:

$L = - \sum y_i \log(\hat{y}_i)$

Hinge Loss：常用于 SVM。 Hinge Loss: commonly used in SVM.

Huber Loss：结合 MSE 与 MAE，兼具稳定性与鲁棒性。 Huber Loss: combines MSE and MAE, balancing stability and robustness.

3. Gradient Descent

梯度下降是一种寻找函数最小值的迭代优化算法。 Gradient Descent is an iterative optimization algorithm for finding a function's minimum.

基本思想是从随机初始点出发，沿下降最快方向不断更新参数直到收敛。 The basic idea is to start from a random point and iteratively update parameters in the steepest descent direction until convergence.

4. Gradient Descent process

梯度下降的数学公式如下： The gradient descent formula is:

$x' = x - \eta \nabla f(x)$

其中 x 是当前参数，η 是学习率，∇f(x) 是梯度。 Where x is the current parameter, η is the learning rate, and ∇f(x) is the gradient.

每次更新方向是下降最快的方向，即梯度的反方向。 Each update is in the direction of the steepest descent, the negative gradient.

5. Relationship between Gradient Descent and Loss Function

梯度下降用于最小化损失函数。 Gradient Descent is used to minimize the loss function.

损失函数是优化目标，梯度是其关于模型参数的导数。 The loss function is the optimization goal, and the gradient is its derivative with respect to model parameters.

梯度下降通过梯度引导参数向损失最小的方向更新。 Gradient Descent uses the gradient to guide parameter updates toward minimizing loss.

换句话说：梯度提供方向，损失函数定义目标。 In other words: the gradient gives the direction, the loss function defines the objective.

6. The gradient at that node

在计算图中，每个节点代表一个变量或操作。 In a computation graph, each node represents a variable or operation.

某一节点的梯度是输出变量对该节点的偏导数。 The gradient at a node is the partial derivative of the output with respect to that node.

例如，对于函数 f(x, y) = 3x^2 + xy，在点 (3, 4) 处的梯度为： For example, for the function f(x, y) = 3x^2 + xy, the gradient at point (3, 4) is:

∇f(3, 4) = [22, 3]

7. Backpropagation

反向传播是一种高效计算神经网络中梯度的方法。 Backpropagation is an efficient method to compute gradients in neural networks.

它基于链式法则，从输出开始，按反方向传播梯度。 It is based on the chain rule, propagating gradients backward from the output.

反向传播将中间结果缓存，避免重复计算。 Backpropagation caches intermediate results to avoid redundant computation.

它适用于有向无环图的计算图结构。 It applies to computation graphs that are directed acyclic graphs (DAGs).

8. The backpropagation computation process

例子： Example:

e = (a + b)(b + 1)，其中 a=2，b=3

正向计算： Forward computation:

c = a + b = 5

d = b + 1 = 4

e = c * d = 20

反向传播： Backpropagation:

∂e/∂c = d = 4

$\partial e/\partial d = c = 5$

$\partial e/\partial a = \partial e/\partial c * \partial c/\partial a = 4$

$\partial e/\partial b = \partial e/\partial c * \partial c/\partial b + \partial e/\partial d * \partial d/\partial b = 9$

总结：每条路径都应用链式法则，若有多条路径通向同一个变量，其梯度需累加。
Summary: Each path applies the chain rule, and if multiple paths lead to a variable, their gradients are accumulated.

二、linear classifier

1. linear classifier

线性分类器是一种基于线性函数将数据划分为不同类别的模型。 A linear classifier is a model that separates data into different classes using a linear function.

它的目标是使用线性决策边界（如直线或超平面）来区分不同类别。 Its goal is to use a linear decision boundary (like a line or hyperplane) to distinguish different classes.

在线性分类器中，预测通常由一个线性函数决定： In a linear classifier, the prediction is usually determined by a linear function.

若 $z > 0$，属于一类；否则属于另一类。 If $z > 0$, it belongs to one class; otherwise to another.

2. linear classifier for 2D data

在二维数据中，线性分类器试图用一条直线将两个类别分开； In 2D data, a linear classifier tries to separate two classes using a straight line.

例如在鸢尾花分类中，使用花瓣长度和宽度为两个维度，将 Iris setosa 与 Iris versicolor 用一条直线区分； For example, in iris classification, petal length and width can be used to draw a line separating Iris setosa from Iris versicolor.

如果该直线可以清晰地区分两类，我们称其为线性可分。 If this line clearly separates the two classes, we call it linearly separable.

3. a linear binary classifier for n-D data and two classes

在线性二分类器中，超平面是线性决策边界； In linear binary classification, the hyperplane is the linear decision boundary.

超平面将整个 n 维空间分为两个部分，对应两个类别； The hyperplane divides the n-dimensional space into two parts, each corresponding to a class.

判定规则如下：若 $w^T x + b > 0$，则预测为一类；否则为另一类； The rule is: if $w^T x + b > 0$, predict one class; otherwise, the other.

w 是法向量，决定超平面方向；b 控制其偏移。 w is the normal vector determining the hyperplane's direction; b controls its offset.

4. a linear multi-class classifier

当类别数 $K > 2$ 时，线性多分类器为每个类别定义一个独立的超平面； When the number of classes $K > 2$, a linear multi-class classifier defines an independent hyperplane for each class.

对于每一个类别 c，计算其置信值 $z_c = w_c^T x + b_c$； For each class c, it computes a confidence score $z_c = w_c^T x + b_c$.

预测结果为最大置信度所对应的类别。 The predicted result is the class with the highest confidence score.

5. a linear multi-class classifier for K classes

将所有类别的线性函数组合成矩阵形式： All class linear functions are combined into a matrix form:

$z = Wx + b$

$z \in \mathbb{R}^K$ 表示所有类别的置信度；$W \in \mathbb{R}^{K \times n}$ 为权重矩阵；$b \in \mathbb{R}^K$ 为偏置。 $z \in \mathbb{R}^K$ represents the confidence scores; $W \in \mathbb{R}^{K \times n}$ is the weight matrix; $b \in \mathbb{R}^K$ is the bias vector.

最终预测为置信值最高的类别。 The final prediction is the class with the highest confidence value.

6. Softmax

Softmax 函数用于将一组任意实数（如置信值）转换为一个概率分布。 The Softmax function is used to convert a set of real numbers (e.g., confidence scores) into a probability distribution.

它的输出在 [0,1] 区间，并且总和为 1。 Its output ranges between [0,1], and the values sum up to 1.

Softmax 公式： Softmax formula:

置信值越高，对应的概率也越高。 The higher the confidence score, the higher the corresponding probability.

常用于多分类问题中，输出每类的概率。 It is commonly used in multi-class problems to output class probabilities.

7. an activation function

激活函数用于引入非线性，使模型能拟合复杂模式。 An activation function introduces non-linearity, enabling the model to fit complex patterns.

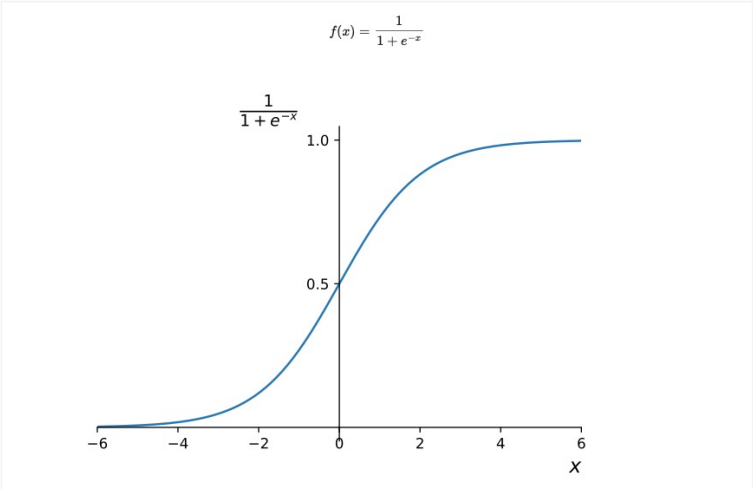
它通常应用在神经网络的每一层输出上。 It is typically applied to the output of each neural network layer.

没有激活函数，网络本质仍是线性模型。 Without activation functions, the network essentially remains a linear model.

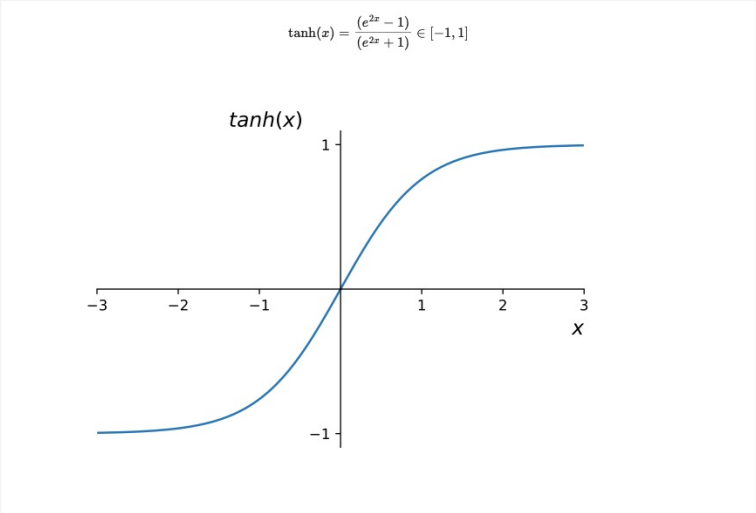
常见激活函数有 sigmoid、tanh、ReLU 等。 Common activation functions include sigmoid, tanh, and ReLU.

8. the logistic (sigmoid) function, tanh, and ReLU

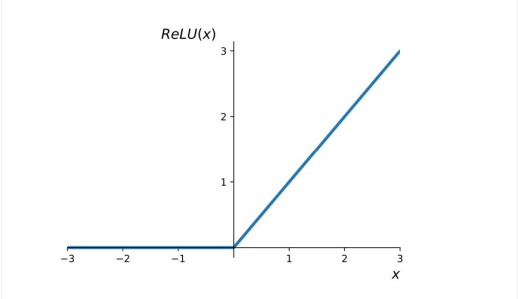
Sigmoid 输出范围为 (0,1)，常用于概率输出。 Sigmoid outputs values in (0,1), often used for probability output.



tanh 输出范围为 (-1,1)，在对称性和梯度方面更稳定。 tanh outputs values in (-1,1), more stable in terms of symmetry and gradient.



ReLU 输出非负值，计算效率高，常用于深度神经网络。 ReLU outputs non-negative values, is computationally efficient, and is commonly used in deep neural networks.



三、measuring classifier performance

3.1 Confusion matrix

A simple and informative way of setting out the performance of a multiclass classifier is with a confusion matrix that sets out the number of correct predictions and the number of incorrect predictions (confusions) as a matrix such as this:

		Predicted class		
		Car	Bicycle	Truck
Actual class	Car	8	2	1
	Bicycle	1	7	2
	Truck	2	0	12

In this example, eight cars were predicted correctly and two cars were incorrectly predicted to be bicycles.

3.2 Accuracy

From the confusion matrix we can compute accuracy, the proportion of correct predictions and error rate, the proportion of incorrect predictions. Note that:

$$accuracy = 1 - error\ rate$$

From the confusion matrix above we have:

$$accuracy = \frac{\#(correct\ predictions)}{\#(dataset)} = \frac{27}{35}$$

$$error\ rate = \frac{\#(incorrect\ predictions)}{\#(dataset)} = \frac{8}{35}$$

真正率（True Positive Rate, TPR） 是指被正确预测为正类的实际正类样本所占的比例。在“有缺陷的零件”这个例子中，TPR 表示被正确识别出来的有缺陷零件的比例。
假正率（False Positive Rate, FPR） 是指被错误预测为正类的实际负类样本所占的比例。可以将其理解为“误报率”。在检测有缺陷零件的例子中，FPR 是指那些实际完好的零件被错误分类为有缺陷的比例。
真正率也被称为 灵敏度（Sensitivity）。
TPR 和 FPR 之间存在权衡关系，这取决于我们将某样本判为正类时所设定的严格程度。在分类中，我们通常选择概率最高的类别作为预测结果，对于二分类问题，这意味着选择概率大于 0.5 的类别（如果两个类别概率相等，可以任选其一）。但我们也可以设定一个更高的阈值来判定为正类，要求模型在做出“正类”预测前具有更高的置信度。
这样做是为了考虑错误预测可能带来的后果，从而在 TPR 和 FPR 之间取得更合理的平衡。提高阈值可以减少误报（即降低 FPR，这是好事），但也可能漏掉更多真正的正类（即降低 TPR，这是坏事）。
我们可以通过不断调整阈值，绘制出 TPR 与 FPR 之间的变化曲线来可视化这种权衡关系。这条曲线被称为 接收者操作特征曲线（Receiver Operating Characteristic curve，简称 ROC 曲线）。

To emphasise the relative importance of the two classes, we normally refer to the class we are interested in as the positive class and the other one as the negative class. The different outcomes corresponding to each cell in the confusion matrix are then referred to as indicated below:

		Predicted class	
		Positive	Negative
Actual class	Positive	TP = #(true positives)	FN = #(false negatives)
	Negative	FP = #(false positives)	TN = #(true negatives)

Two ways of measuring the performance of binary classifiers are widely used.

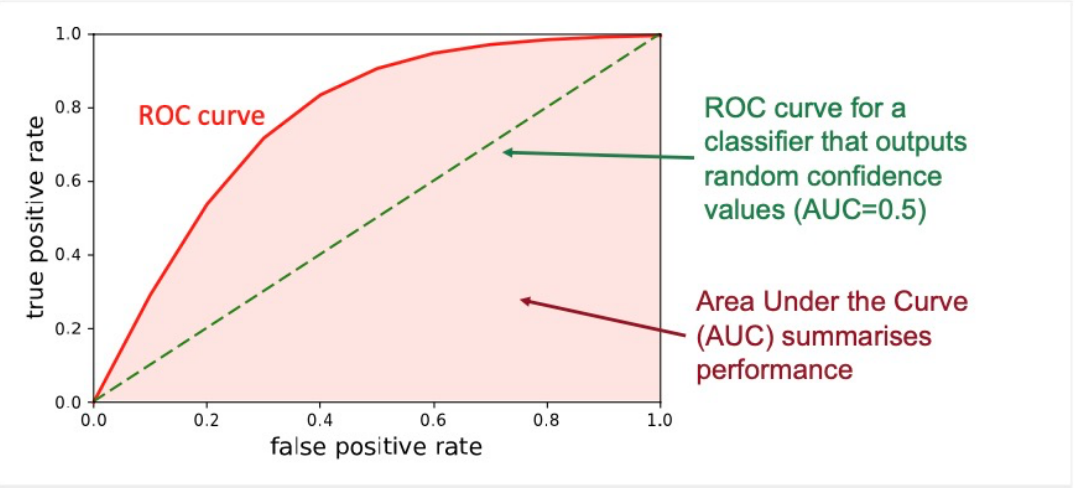


Figure 1. ROC curve.
View image description

Recall 所有真的里面被预测正确的真的 TP/(TP + FN)

Precision 被预测为真的里面实际的真的 TP/(TP + FP)

3.4 Precision and recall in multiclass classification

Precision and recall are also used as a performance measure with respect to individual classes in multi-class classification. In the three class example at the start, we can compute recall and precision values for the truck class just as we did for the positive class in binary classification:

$$recall\ for\ truck = \frac{\#(correctly\ predicted\ trucks)}{\#(actual\ trucks)}$$

$$precision\ for\ truck = \frac{\#(correctly\ predicted\ trucks)}{\#(predicted\ trucks)}$$

These statistics are particularly useful when we have an unbalanced dataset, with very different numbers of examples in each class. Consider for example the following confusion matrix:

		Predicted class		
		Car	Bicycle	Truck
Actual class	Car	1000	5	5
	Bicycle	10	0	0
	Truck	10	0	0

In this case there are many more cars than trucks and bicycles. The classifier makes the wrong prediction for every truck and bicycle, yet the accuracy is high:

$$accuracy = \frac{1000}{1030} = 0.97$$

In this case, the accuracy is misleading. Fortunately the recall for the truck and bicycle classes show that the performance on these two classes is poor.

$$recall\ for\ car = \frac{1000}{1010} = 0.99$$

$$recall\ for\ truck = \frac{0}{10} = 0$$

$$recall\ for\ bicycle = \frac{0}{10} = 0$$

$$F = \frac{2}{precision^{-1} + recall^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

四、multilayer networks

1. a Multilayer Network

多层网络是指由多个线性层和非线性激活函数交替堆叠构成的神经网络结构，常被称为多层感知器（MLP）。 A multilayer network refers to a neural network structure composed of alternating linear layers and nonlinear activation functions, commonly known as a multilayer perceptron (MLP).

每一层都是一个仿射变换，形式为： Each layer is an affine transformation in the form of:

线性变换后会使用激活函数引入非线性。 After the linear transformation, an activation function is used to introduce non-linearity.

多层结构可以学习和表示复杂的非线性特征。 The multilayer structure can learn and represent complex nonlinear features.

每一层的输出可以看作是下一层的输入。 The output of each layer can be viewed as the input to the next layer.

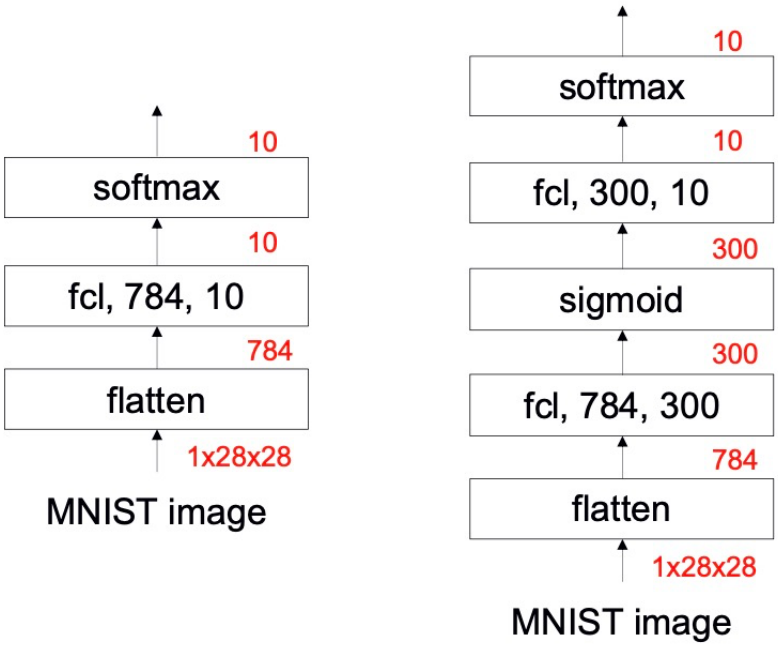
在 PyTorch、TensorFlow 中称为 Linear layer，在 Keras 中称为 Dense layer，图示中常标记为 fcl（fully connected layer）。 In PyTorch and TensorFlow it is called a Linear layer, in Keras a Dense layer, and often labeled as fcl in diagrams.

训练流程通常包括： The training process usually includes:

使用训练集拟合模型； Fitting the model using the training set;

验证集评估泛化能力； Evaluating generalization ability with the validation set;

测试集获得最终性能。 Obtaining final performance on the test set.



2. Stochastic Gradient Descent (SGD)

随机梯度下降是一种对损失函数进行最小化的优化算法。 Stochastic Gradient Descent is an optimization algorithm for minimizing the loss function.

SGD 每次仅使用一个小批量的样本计算梯度并更新参数。 SGD computes gradients and updates parameters using only a small batch of samples each time.

每次迭代仅使用部分样本，使得计算更高效，占用内存更少。 Each iteration uses only a subset of samples, making computation more efficient and memory-efficient.

更新更加频繁，能更快地逼近最优参数。 The updates are more frequent, allowing faster convergence to optimal parameters.

由于数据存在随机性，优化路径中带有“噪声”，反而有利于跳出局部最优，提升模型泛化能力。 Due to data randomness, the noisy optimization path can help escape local minima and improve generalization.

使用小批量训练可以并行加速，例如 GPU 支持批量操作。 Using mini-batch training can accelerate via parallelization, such as GPU batch support.

常见 minibatch 大小：8、16、32、64 等。 Common mini-batch sizes include: 8, 16, 32, 64, etc.

3. momentum in Gradient Descent

动量是对标准梯度下降的改进，它将之前的梯度历史考虑在内，使参数更新更具惯性和平滑性。 Momentum is an improvement to standard gradient descent that incorporates past gradients to smooth and stabilize updates.

直观理解为：参数在损失函数曲面上像小球一样运动，会有惯性。 Intuitively, parameters move on the loss surface like a ball with momentum.

更新公式中加入速度项 v，并叠加之前的方向。 The update formula includes a velocity term v, incorporating the previous direction.

能加速收敛、减缓震荡，尤其在鞍点附近更稳定。 It accelerates convergence and reduces oscillation, especially around saddle points.

Nesterov 动量是动量的改进版，具有更好的前瞻性。 Nesterov momentum is an improved version of momentum with better foresight.

其中 β 是动量因子，α 是学习率，∇θf(θ) 是当前梯度。 Here, β is the momentum factor, α is the learning rate, and ∇θf(θ) is the current gradient.

五、convolution

1. convolution

卷积是一种图像处理或信号处理操作，广泛应用于图像平滑、边缘检测和深度学习等领域。 Convolution is an operation in image or signal processing, widely used in smoothing, edge detection, and deep learning.

卷积操作通过一个称为 kernel（核）或 mask（掩码）的小矩阵，对图像的每个像素邻域进行加权求和。 The convolution operation uses a small matrix called a kernel or mask to perform weighted sums on each pixel neighborhood.

核通常滑动整张图像，每次移动一步，对应位置的像素和核的值逐元素相乘再求和。 The kernel slides over the image, multiplying and summing values with overlapping pixels.

当核在应用前旋转 180° 后再进行加权求和，这一过程才被严格定义为卷积。 When the kernel is rotated 180° before applying the sum, the process is strictly defined as convolution.

若核具有旋转对称性，则卷积和交叉相关结果相同。 If the kernel is rotation symmetric, the result of convolution and cross-correlation is the same.

卷积的本质是局部信息加权组合，输出图像每个像素的新值反映其周围邻域的信息。 Convolution essentially combines local information, and the new pixel value reflects neighborhood information.

2. padding, stride, dilation in convolution

Padding（填充）、Stride（步长）和 Dilation（扩张）是卷积操作中三个重要参数。 Padding, stride, and dilation are three important parameters in convolution operations.

Padding 是在图像边缘添加额外的行或列（通常为 0）。 Padding adds extra rows or columns around the image, usually with zeros.

其目的是保持输出图像大小与原图一致。 Its purpose is to keep the output image the same size as the input.

Stride 是每次移动核时的步长。 Stride is the number of pixels the kernel moves at each step.

若 stride 大于 1，则输出图像更小，特征图尺寸下降更快。 If stride is greater than 1, the output image is smaller and the feature map shrinks faster.

Dilation 是在核元素之间插入间隔，使感受野更大。 Dilation inserts spacing between kernel elements to enlarge the receptive field.

dilation = 1 表示核连续无间隔；dilation = 2 表示核之间隔一个像素采样。 Dilation = 1 means contiguous kernel; dilation = 2 means sampling with one pixel interval.

可在不增大 kernel 大小的情况下扩大感知范围。 It allows increasing the receptive field without enlarging the kernel size.

常用于空洞卷积。 Often used in dilated convolutions.

3. Gaussian smoothing

高斯平滑是一种利用高斯分布加权图像邻域像素的图像平滑方法，用以去除小尺度噪声或细节。 Gaussian smoothing is a method of image smoothing that weights neighboring pixels based on a Gaussian distribution to remove small-scale noise or detail.

Gaussian kernel 以图像中心像素为核心，对周围像素施加逐渐减弱的权重。 The Gaussian kernel applies gradually decreasing weights to pixels around the central pixel.

核大小可为 3x3、5x5、甚至 19x19。 Kernel size can be 3x3, 5x5, or even 19x19.

σ 控制高斯分布的扩展程度（即模糊程度）。 σ controls the spread of the Gaussian distribution (i.e., the degree of blurring).

σ 小，平滑弱，保留更多细节；σ 大，平滑强，细节被抹去。 Small σ results in weak smoothing and more detail; large σ results in strong smoothing and reduced detail.

4. Demonstrate the application of Gaussian smoothing

高斯平滑的应用流程如下： The application process of Gaussian smoothing is as follows:

构建一个以 0 为中心的二维整数网格（如 -2 至 2）。 Construct a 2D integer grid centered at 0 (e.g., -2 to 2).

在该网格上用公式计算每个点的权重。 Compute the weight of each point on the grid using the formula.

将所有权重归一化，使其总和为 1。 Normalize all weights so their sum equals 1.

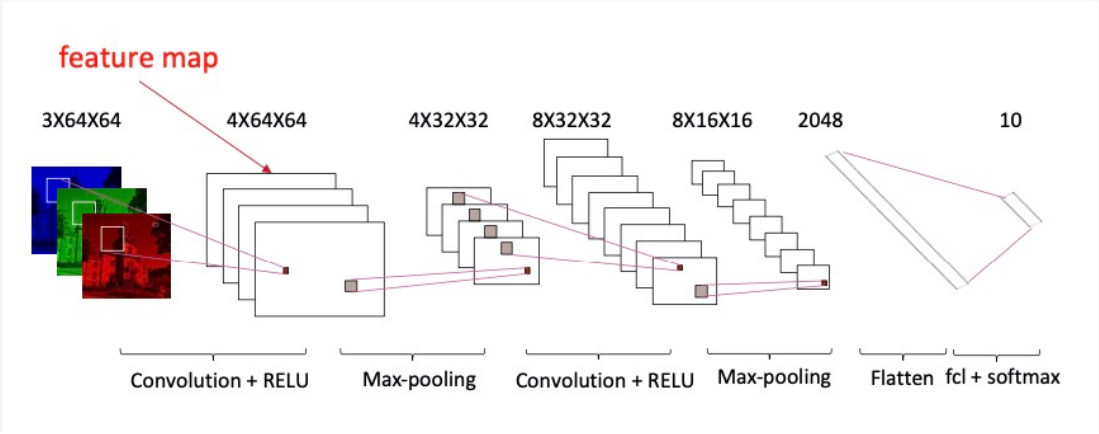
用该核与图像做卷积，得到平滑图像。 Convolve the kernel with the image to obtain the smoothed result.

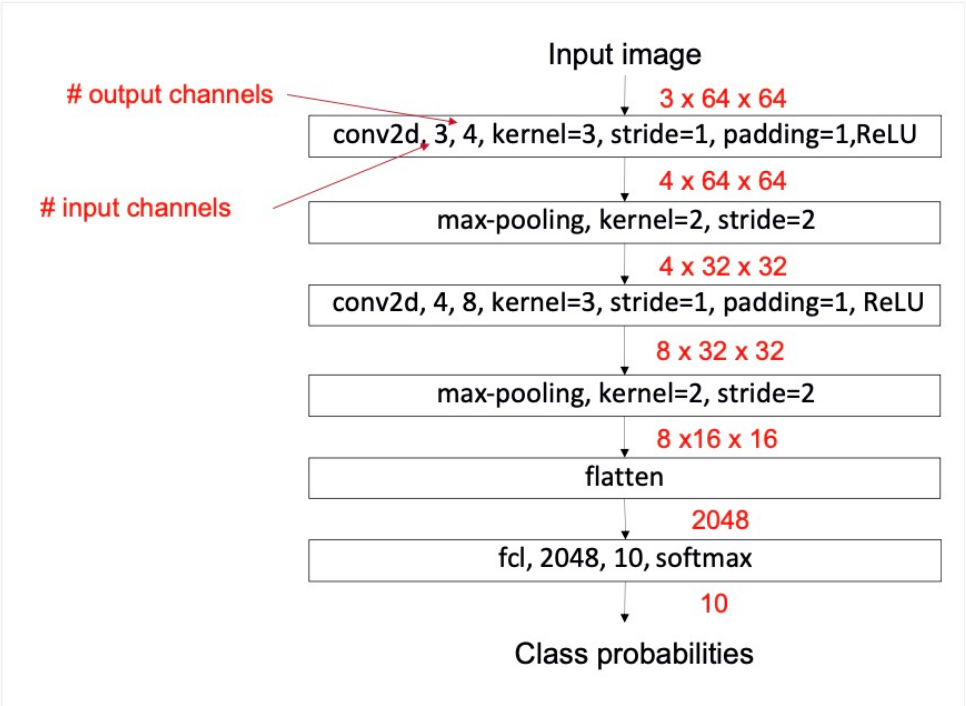
图像边缘和细节变得更加模糊。 Edges and details in the image become more blurred.

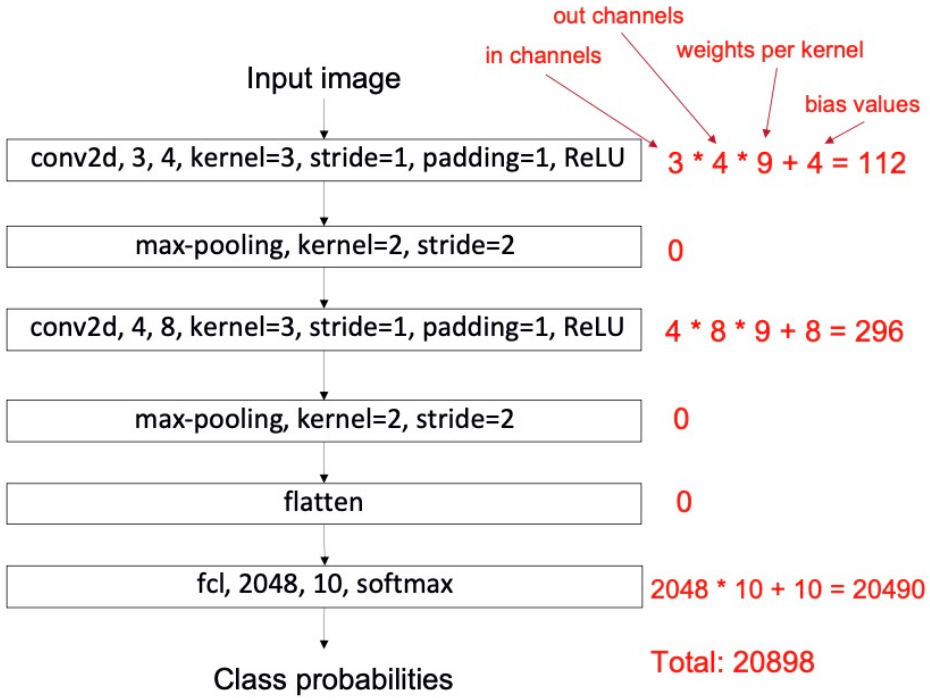
可以有效去除图像中的高频噪声。 High-frequency noise in the image can be effectively removed.

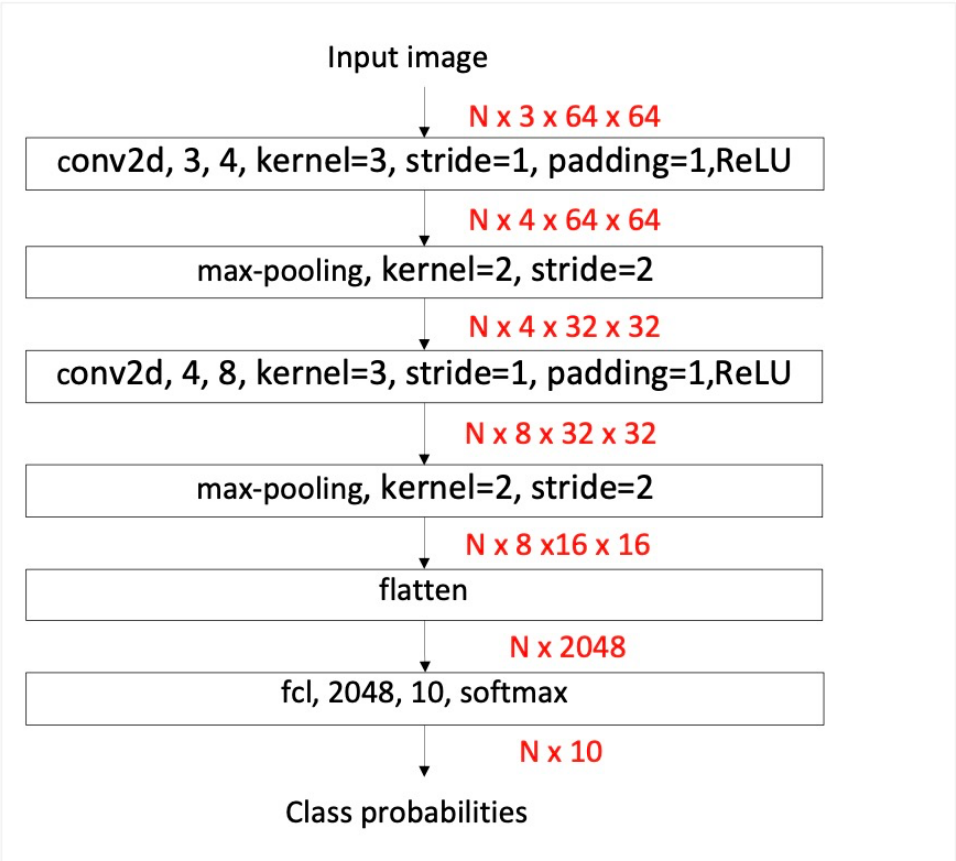
实际图像处理工具如 OpenCV、PIL、TensorFlow/Keras、PyTorch 中都集成了高斯平滑方法。 Practical tools like OpenCV, PIL, TensorFlow/Keras, and PyTorch include built-in Gaussian smoothing methods.

六、Convolutional neural networks




Input image
output channels
3 x 64 x 64
conv2d, 3, 4, kernel=3, stride=1, padding=1,ReLU
input channels
4 x 64 x 64
max-pooling, kernel=2, stride=2
4 x 32 x 32
conv2d, 4, 8, kernel=3, stride=1, padding=1, ReLU
8 x 32 x 32
max-pooling, kernel=2, stride=2
8 x16 x 16
flatten
2048
fcl, 2048, 10, softmax
10
Class probabilities


out channels
in channels
weights per kernel
bias values
Input image
conv2d, 3, 4, kernel=3, stride=1, padding=1, ReLU
3 * 4 * 9 + 4 = 112
max-pooling, kernel=2, stride=2
0
conv2d, 4, 8, kernel=3, stride=1, padding=1, ReLU
4 * 8 * 9 + 8 = 296
max-pooling, kernel=2, stride=2
0
flatten
0
fcl, 2048, 10, softmax
2048 * 10 + 10 = 20490
Class probabilities
Total: 20898



卷积公式（无 dilation）：

$$\text{Output size} = \left\lfloor \frac{\text{Input size} + 2 \times \text{padding} - \text{kernel size}}{\text{stride}} \right\rfloor + 1$$

我们设：

- kernel size = 2
- stride = 1
- output size = input size

代入公式可得：

$$\text{Input} + 2p - 2 + 1 = \text{Input} \Rightarrow 2p - 1 = 0 \Rightarrow p = 0.5$$

1. How to deal with over-fitting

降低模型复杂度：减少模型的参数数量或层数； Reduce model complexity: decrease the number of parameters or layers.

数据增强：通过对原始图像进行旋转、缩放、翻转、亮度调整等操作，扩展训练集；Data augmentation: expand the training set by rotating, scaling, flipping, and adjusting brightness of images.

使用 Dropout：在训练过程中随机屏蔽部分神经元，防止模型过度依赖某些特征；Use Dropout: randomly deactivate neurons during training to avoid overfitting.

应用 Batch Normalisation：通过规范化激活值减小内部协变量偏移，帮助模型更稳定训练； Apply Batch Normalisation: normalize activations to reduce internal covariate shift.

提前停止（Early Stopping）：当验证集性能不再提升时停止训练； Early stopping: stop training when validation performance stops improving.

增加训练数据量：提高模型泛化能力。 Increase training data volume: improve the generalization ability of the model.

2. How to Reduce Model complexity

减少网络的层数或每层的神经元数； Reduce the number of layers or neurons per layer.

使用更小的卷积核（例如从 5×5 改为 3×3）； Use smaller convolution kernels (e.g., from 5×5 to 3×3).

减少特征图通道数； Reduce the number of feature map channels.

使用参数共享（如卷积）而非全连接； Use parameter sharing (e.g., convolution) instead of full connection.

在实际任务中还可以使用剪枝、量化等技术。 In practice, pruning and quantization techniques can also be used.

3. Data augmentation

数据增强是一种通过对训练样本进行随机变换生成新样本的方法，增强数据多样性，减轻过拟合。 Data augmentation is a method of generating new samples by randomly transforming training data to enhance diversity and reduce overfitting.

常见方法包括： Common methods include:

图像旋转、平移、缩放、翻转； Image rotation, translation, scaling, flipping;

改变图像亮度、饱和度、色调； Adjusting image brightness, saturation, and hue;

模拟摄像机位置变化和类别内差异； Simulating camera position variations and intra-class diversity;

所有变换样本继承原始标签，训练数据量大幅增加。 All transformed samples retain the original label, significantly increasing training data volume.

4. Dropout

Dropout 是一种正则化方法： Dropout is a regularization method:

在训练过程中，以概率 p 将某些神经元的输出设为 0； During training, with probability p, some neuron outputs are set to 0;

强迫网络不依赖单一神经元，提升泛化能力； Forces the network not to rely on individual neurons, improving generalization.

在前向传播时，会对保留的神经元输出按比例缩放，保持整体输出期望不变； During forward pass, outputs of retained neurons are scaled to maintain expected output.

只在训练模式使用，评估时不启用； Used only in training mode, not in evaluation.

本质是减少 co-adaptation，提升鲁棒性。 Essentially reduces co-adaptation and improves robustness.

5. Batch Normalisation

Batch Normalisation 是在每个小批量内对激活值进行标准化操作的过程。 Batch Normalisation is the process of normalizing activations within each mini-batch.

每个通道内的值都会基于其均值与标准差被标准化； Values in each channel are normalized based on mean and standard deviation;

然后再乘以一个可学习的缩放因子 α，加上偏移量 β； Then scaled by a learnable factor α and shifted by a bias β;

优点包括： Advantages include:

减少梯度消失/爆炸； Reducing vanishing/exploding gradients;

加速收敛； Faster convergence;

具有轻微正则化效果； Provides mild regularization effect;

训练时使用 batch 统计量，测试时使用训练时记录的统计量。 Uses batch statistics during training, and recorded statistics during testing.

6. Receptive Field

感受野是指卷积网络中某个神经元对应原始输入图像的区域大小。 Receptive field refers to the size of the region in the original input image that a neuron in a convolutional network corresponds to.

每个特征图位置是输入图像某一块区域的处理结果； Each feature map position represents the processed result of a certain region in the input image;

该区域就是这个位置的感受野； This region is the receptive field of that position;

随着卷积层和池化层叠加，感受野会不断增大。 With more convolution and pooling layers, the receptive field keeps increasing.

7. Why is Receptive Field important

感受野的重要性： Importance of receptive field:

影响模型的空间上下文捕捉能力； It affects the model's ability to capture spatial context;

感受野太小 → 难以获取全局信息； Too small receptive field → cannot capture global context;

感受野太大 → 对局部细节不敏感； Too large receptive field → less sensitivity to local details;

实际应用中需保证感受野能覆盖目标物体区域； In practice, ensure the receptive field covers the target object;

控制感受野是设计 CNN 架构的关键。 Controlling receptive field is key in CNN architecture design.

8. How to calculate Receptive Field

感受野大小随层级传播，从输入向输出累加； Receptive field size accumulates through the layers from input to output;

若第一层使用 3×3 卷积，则初始感受野为 3×3； If the first layer uses a 3×3 convolution, the initial receptive field is 3×3;

第二层使用 4×4 卷积 → 感受野变为 6×6； Second layer with 4×4 convolution → receptive field becomes 6×6;

再加上一个 2×2 池化（stride=2） → 感受野为 7×7； With a 2×2 pooling (stride=2) → receptive field becomes 7×7;

总体计算考虑卷积核大小、stride、padding 等因素； Overall computation considers kernel size, stride, padding, etc.;

每层都会“扩大”前一层的感受野。 Each layer expands the receptive field from the previous layer.

符号	含义
k	卷积核大小（kernel size）
s	步长（stride）
p	padding（补零）
r	当前层的感受野大小
j	当前层的 jump（实际步长，在输入图上的跳跃）

对于第 l 层（从输入往后推）：

- 感受野大小：

$$r_l = r_{l-1} + (k_l - 1) \cdot j_{l-1}$$

- jump（在输入上的步幅）：

$$j_l = j_{l-1} \cdot s_l$$

初始条件（输入层）：

- $r_0 = 1$（一个像素）
- $j_0 = 1$

一个简单的 CNN 网络如下：

层次	类型	kernel	stride	padding
1	Conv	3	1	1
2	MaxPool	2	2	0
3	Conv	3	1	1

我们来手算第 3 层某个神经元的感受野：

1. **第0层（输入）**
   - 感受野：1，jump：1
2. **第1层（Conv3×3, s=1, p=1）**
   - 感受野：$1 + (3-1) \cdot 1 = 3$
   - jump：$1 \cdot 1 = 1$
3. **第2层（MaxPool2×2, s=2, p=0）**
   - 感受野：$3 + (2-1) \cdot 1 = 4$
   - jump：$1 \cdot 2 = 2$
4. **第3层（Conv3×3, s=1, p=1）**
   - 感受野：$4 + (3-1) \cdot 2 = 8$
   - jump：$2 \cdot 1 = 2$

9. Grad-Cam

Grad-CAM 是一种用于可视化 CNN 分类决策依据的技术。 Grad-CAM is a technique for visualizing the reasoning behind CNN classification decisions.

目标：找出输入图像中最影响分类结果的区域； Goal: identify the regions in the input image that most affect classification output;

具体方法： Method:

对一张图像运行模型，选择感兴趣的类别； Run the model on an image and choose the target class;

计算该类别置信度对卷积层激活图的梯度； Compute gradients of the class confidence with respect to feature maps;

对每个特征图求该梯度的平均值，作为权重； Average the gradients to get weights for each feature map;

用这些权重对原特征图加权求和； Weight the feature maps and sum them up;

应用 ReLU，得到 Grad-CAM 热力图； Apply ReLU to obtain the Grad-CAM heatmap;

结果图可以叠加在原图上，可解释模型关注区域。 The result can be overlaid on the original image to show where the model is focusing.

七、Fine Tuning

1. Transfer Learning

迁移学习是一种深度学习策略，旨在将已有模型从一个任务/数据域迁移到另一个任务/数据域上。 Transfer Learning is a deep learning strategy that aims to transfer a model from one task/domain to another.

目的：利用在大数据集（如 ImageNet）上训练好的模型，适应新的小数据集（如医疗图像）。 Purpose: Leverage models pre-trained on large datasets (e.g., ImageNet) to adapt to new small datasets (e.g., medical images).

优势： Advantages:

避免从零开始训练，节省计算资源； Avoid training from scratch and save computational resources;

提高在小数据集上的表现； Improve performance on small datasets;

迁移方式的变化： Transfer modes:

相同领域，不同任务（如从狗的分类迁移到狗的检测）； Same domain, different task (e.g., from dog classification to dog detection);

不同领域，相同任务（如从自然图像迁移到医学图像分类）。 Different domain, same task (e.g., from natural to medical image classification).

2. Fine Tuning

微调是迁移学习中的一种具体策略，通过对预训练模型进行小范围的再训练来适配目标任务。 Fine tuning is a strategy in transfer learning where pre-trained models are slightly retrained to fit the target task.

对已有模型的部分层进行再训练； Retrain parts of the existing model;

保留通用特征（如边缘、颜色等）； Retain general features (e.g., edges, colors);

仅调整与新任务相关的部分（如分类器）； Only adjust parts related to the new task (e.g., classifier);

可以选择冻结部分层（不更新权重）、重新训练其他层。 Can choose to freeze some layers (do not update weights) and retrain others.

3. How to do Fine Tuning

根据数据量和任务的差异，有多种微调方法： There are several fine-tuning methods depending on data size and task differences:

方法一：加载预训练模型并重新训练整个网络 Method 1: Load a pre-trained model and retrain the entire network

适合数据量充足； Suitable for large datasets;

初始化为预训练权重，训练时从头优化所有参数； Initialize with pre-trained weights and optimize all parameters from scratch;

计算成本较高，但适用于与原任务差异较大的目标。 High computational cost but useful when the target task is quite different from the original.

方法二：冻结卷积基，仅训练最后的全连接层 Method 2: Freeze the convolutional base and only train the final fully connected layer

适合小数据集、计算资源有限的情况； Suitable for small datasets or limited computational resources;

卷积基提取的低层次特征（如边缘）对多数任务通用； Low-level features from convolutional base are general across tasks;

仅优化最后一两层分类器。 Only optimize the final one or two classifier layers.

方法三：冻结部分卷积块，训练其余卷积层与全连接层 Method 3: Freeze some convolution blocks and train the rest plus fully connected layers

平衡计算量与特征适应性； Balance between computation and feature adaptability;

通常冻结越低层的网络结构，越能保留通用特征； Freezing lower layers retains more general features;

高层特征对具体任务更敏感，可重新训练以提高适应性。 High-level features are more task-specific and can be fine-tuned to improve adaptation.

七、image segmentation

1. Semantic Segmentation

语义分割是一种计算机视觉任务，目标是为图像中的每一个像素赋予一个对象类别标签。 Semantic segmentation is a computer vision task that assigns an object class label to each pixel in an image.

与图像分类不同，语义分割是逐像素的分类任务； Unlike image classification, semantic segmentation is a pixel-level classification task;

每个像素都被分配一个类标签； Each pixel is assigned a class label;

输出通常是与输入图像大小相同的掩码，每个像素值表示其所属类别； The output is usually a mask of the same size as the input image, where each pixel value indicates its class;

性能评估通常使用 Jaccard index（即 IoU）。 Performance is often evaluated using the Jaccard index (Intersection over Union, IoU).

2. Patch-wise Classification

Patch-wise 分类是早期语义分割方法之一，将图像分为小块，对每个小 patch 的中心像素进行分类。 Patch-wise classification is an early method for semantic segmentation, dividing images into patches and classifying the center pixel of each.

使用 CNN 对每个像素周围的图像 patch 进行分类； CNN is used to classify each pixel based on its surrounding patch;

每个 patch 的标签是中心像素的类别； Each patch is labeled with the class of its center pixel;

可将任务转换为图像分类任务进行训练。 This approach allows training using standard image classification pipelines.

3. pros and cons of Patch-wise Classification

优点： Pros:

实现简单，容易将图像分类模型迁移用于分割； Simple to implement, easy to adapt image classification models for segmentation;

适用于分辨率较小或结构单一的图像。 Suitable for low-resolution or structurally simple images.

缺点： Cons:

推理速度慢，需逐个像素生成 patch； Inference is slow since patches must be generated for each pixel;

忽略了全图上下文信息； Ignores global image context;

模型训练效率低，冗余计算多； Inefficient training with redundant computation;

输出分辨率可能较低（若使用池化）。 Output resolution may be low (if pooling is used).

4. Encoder-Decoder Architecture

Encoder-Decoder 架构是一种用于语义分割的深度学习框架，由两个部分组成： The encoder-decoder architecture is a deep learning framework for semantic segmentation consisting of two parts:

Encoder：通过卷积、池化等操作提取图像特征，压缩空间维度； Encoder: extracts image features using convolution and pooling, reducing spatial dimensions;

Decoder：通过上采样恢复图像空间结构，生成分割图； Decoder: restores the image's spatial structure using upsampling to produce segmentation maps;

中间的 embedding layer 作为图像的语义表示； The intermediate embedding layer represents the semantic content of the image;

属于特征学习范式，可捕捉全局上下文信息。 It is a feature learning approach that captures global context information.

5. pros and cons of Encoder-Decoder Architecture

优点： Pros:

可输出与输入图像相同大小的分割图； Can output segmentation maps of the same size as the input image;

能捕捉全局特征，有较强语义表达力； Captures global features and has strong semantic representation capability;

结构灵活，可扩展。 Flexible and scalable architecture.

缺点： Cons:

上采样可能导致图像细节丢失； Upsampling may cause loss of image details;

训练过程中对位置敏感度下降； Position sensitivity may reduce during training;

对高分辨率图像处理效率较低。 Less efficient for high-resolution images.

6. U-Net Architecture

U-Net 是在 Encoder-Decoder 架构基础上发展而来的一种网络结构，最初用于医学图像分割。 U-Net is an architecture based on the encoder-decoder framework, originally designed for medical image segmentation.

U 形结构：编码器下采样、解码器上采样； U-shaped structure: encoder downsamples, decoder upsamples;

关键特征：跳跃连接，将 encoder 每一层的特征直接传给对应 decoder 层； Key feature: skip connections that pass encoder features directly to corresponding decoder layers;

这样做有助于恢复空间信息，提升边缘精度； This helps restore spatial information and improve edge precision;

可用于 2D 或 3D 分割任务。 Applicable to both 2D and 3D segmentation tasks.

7. pros and cons of U-Net Architecture

优点： Pros:

保留局部细节信息（通过跳跃连接）； Preserves local detail via skip connections;

可以在少量训练数据下达到优秀表现； Performs well even with limited training data;

广泛应用于医学图像、遥感、工业检测等领域； Widely used in medical imaging, remote sensing, and industrial inspection;

可拓展性强，可用于多通道、多类别分割。 Highly extensible for multi-channel and multi-class segmentation.

缺点： Cons:

结构复杂，内存占用较大； Complex structure with high memory usage;

对输入大小敏感，需要标准化； Sensitive to input size and requires standardization;

难以直接扩展到多分辨率联合处理。 Challenging to extend directly to multi-resolution fusion.

八、sequential data

1. Sequential Data

顺序数据是指元素之间具有自然顺序或时间先后关系的数据类型。 Sequential data refers to data where elements have a natural or temporal order.

常见的顺序数据包括： Common sequential data includes:

文本（词或字符序列） Text (word or character sequences)

视频（图像帧序列） Videos (sequences of image frames)

医疗记录（按时间顺序排列的诊断、症状、治疗等事件） Medical records (chronological events like diagnoses, symptoms, and treatments)

顺序数据的建模关注如何根据历史推测未来。 The modeling of sequential data focuses on predicting the future based on history.

通常需要使用序列模型（如 RNN、Transformer）建模这种时间相关性。 Sequence models like RNNs or Transformers are typically used to model temporal dependencies.

2. Image Captioning

图像描述生成是指输入一张图片，输出一段自然语言文字描述。 Image captioning refers to generating a natural language description from an image.

输出为一组具有语义连贯性的单词或字符序列； The output is a semantically coherent sequence of words or characters;

本质上是图像到序列的转换任务； Essentially, it is a task of converting images to sequences;

常用于图像理解、辅助盲人阅读、搜索引擎等场景。 It is often used in image understanding, aiding the visually impaired, and search engines.

3. Sentiment Analysis

情感分析是指对一段文本进行分类，判断其表达的是正面、负面或其他情绪倾向。 Sentiment analysis classifies a text to determine if it expresses a positive, negative, or other emotional tone.

输入是评论、评价等自然语言文本； The input is natural language text like reviews or opinions;

输出可以是： The output can be:

二分类（正面/负面） Binary classification (positive/negative)

多级评分（如五星制） Multilevel rating (e.g., 5-star scale)

示例：对电影《小丑》的影评输出“positive”。 Example: A review of the movie 'Joker' outputs 'positive'.

4. Personality Scores from Text

从文本中提取人格评分是指将自然语言文本映射为个体的心理或行为特征评分。 Extracting personality scores from text means mapping natural language to psychological or behavioral traits.

输出为一个包含多个维度的向量，如五大人格特质。 The output is a multi-dimensional vector, such as the Big Five personality traits.

示例：IBM Personality Insights 通过用户输入文本生成 5 个维度的人格得分；
Example: IBM Personality Insights generates five trait scores from input text;

可用于用户画像、招聘筛选、个性化推荐等。 It can be used for user profiling, recruitment filtering, and personalized recommendation.

5. Machine Translation

机器翻译是将一种语言的文本转换为另一种语言的过程，例如从英文翻译为法文。
Machine translation is the process of converting text from one language to another, such as English to French.

输入输出都是序列，属于典型的 Seq2Seq 任务； Both input and output are sequences, making it a typical Seq2Seq task;

示例： Example:

输入："A woman is running with her dog..." Input: "A woman is running with her dog..."

输出："Une femme court avec son chien..." Output: "Une femme court avec son chien..."

6. Text to Speech

文本转语音是指将自然语言文本转换为人类可听的语音波形。 Text-to-speech converts natural language text into human-audible audio waveforms.

输入为文本，输出为音频样本序列； The input is text, and the output is an audio sample sequence;

属于 Seq2Seq 任务（文本序列 → 音频序列）； It is a Seq2Seq task (text sequence to audio sequence);

应用于语音助手、导航系统、无障碍技术等。 Used in voice assistants, navigation systems, and accessibility technologies.

7. Stochastic Processes

随机过程是指随时间演化的、每个时刻的状态是概率分布随机变量的过程。 A stochastic process evolves over time where each state is a random variable with a probability distribution.

在序列建模中，它表示： In sequence modeling, it represents:

已知过去序列，预测当前或下一个值的概率分布； Given past sequences, predict the probability distribution of the current or next value;

可采用马尔可夫假设简化，如 n 阶马尔可夫过程只依赖最近 n 个时间点； It can be simplified using the Markov assumption, such as an n-order Markov process relying on the last n steps;

神经网络可用于学习这些条件概率分布。 Neural networks can be used to learn these conditional probability distributions.

九、recurrent neural networks

1. Recurrent Neural Networks (RNN)

循环神经网络（RNN）是一种专门用于处理序列数据的神经网络结构。 Recurrent Neural Networks (RNNs) are a type of neural network designed for processing sequential data.

与传统前馈神经网络不同，RNN 有隐藏状态，能记住过去的输入； Unlike feedforward neural networks, RNNs have hidden states that retain past inputs;

在每个时间步，RNN 接收当前输入和前一时刻的隐藏状态，更新新的隐藏状态； At each time step, the RNN takes the current input and the previous hidden state to update a new hidden state;

这种“循环结构”使得 RNN 能够学习时间上的依赖关系； This 'recurrent structure' enables RNNs to learn temporal dependencies;

最常见的形式是 Elman RNN（或 vanilla RNN）。 The most common form is the Elman RNN (or vanilla RNN).

2. How does Recurrent Neural Networks generate text

RNN 生成文本的过程如下： The process for RNN to generate text is as follows:

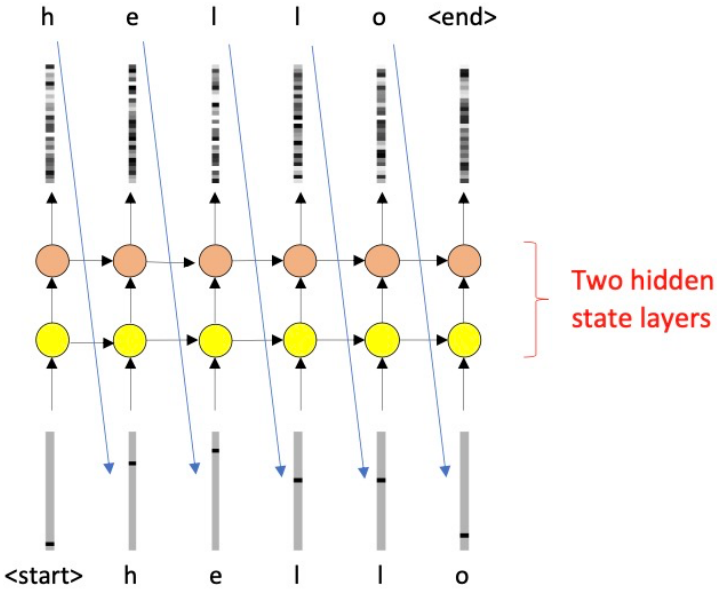
输入文本片段（例如字符或单词序列）； Input a text fragment (e.g., sequence of characters or words);

每次处理一个字符，产生一个概率分布（输出）； At each step, process one character and produce a probability distribution;

从中采样或选取概率最高的下一个字符，作为下一时刻的输入； Sample or select the highest probability character as the next input;

重复该过程直到生成 <end> 标记或达到最大长度； Repeat the process until an <end> token is produced or the maximum length is reached;

可用于自动创作、对话系统、代码生成等。 It can be used in applications such as automatic writing, dialogue systems, and code generation.



3. How to train Recurrent Neural Networks

RNN 的训练过程包括以下步骤： Training an RNN involves the following steps:

采用大量固定长度的字符或词序列作为训练样本； Use a large number of fixed-length character or word sequences as training samples;

将每个样本输入为序列 x1, x2, ..., xT，目标是预测下一个字符 y1, y2, ..., yT； Each sample is input as a sequence x1, x2, ..., xT with the goal to predict the next character y1, y2, ..., yT;

通过反向传播（BPTT）计算梯度； Use backpropagation through time (BPTT) to compute gradients;

使用负对数似然损失函数； Use negative log-likelihood as the loss function;

多个样本组成一个 batch，总损失是所有样本 log-likelihood 之和； Multiple samples form a batch, and the total loss is the sum of all sample log-likelihoods;

通过优化算法（如 SGD 或 Adam）迭代更新参数。 Update parameters iteratively using optimization algorithms such as SGD or Adam.

4. How to calculate RNN parameters

RNN 的主要参数包括以下矩阵和向量： The main parameters in RNN include the following matrices and vectors:

W：隐藏状态之间的连接（hidden-to-hidden） W: connections between hidden states (hidden-to-hidden)

U：输入与隐藏层之间的连接（input-to-hidden） U: input-to-hidden layer connections

V：隐藏层与输出之间的连接（hidden-to-output） V: hidden-to-output connections

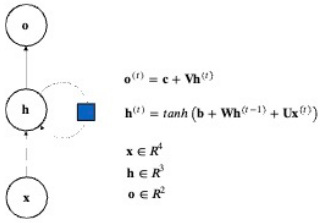
b：隐藏层偏置 b: bias for the hidden layer

c：输出层偏置 c: bias for the output layer

参数总数依赖于输入维度、隐藏层维度和输出维度。 The total number of parameters depends on input, hidden, and output dimensions.

Question 4

Consider the RNN below, defined by the given functions and the dimensionalities of the input, output and hidden layer. The connection with the square box represents the recurrent connection of the hidden vector h. How many parameters are there in this RNN (i.e. weights and bias values)?



(A) 30
(B) 300
(C) 32
(D) 320
(E) 24

out of 4

[Question 4 Total: 4 marks]

名称	说明	大小
U	输入权重矩阵（作用于 x）	$\mathbb{R}^{3\times4}$（从 4 输入 → 3 隐藏）
W	隐藏状态的循环权重（作用于 $h^{(t-1)}$）	$\mathbb{R}^{3\times3}$
b	隐藏层偏置项	长度为 3
V	输出层权重矩阵（作用于隐藏层）	$\mathbb{R}^{2\times3}$
c	输出层偏置项	长度为 2

1.U: 3x4=12

2.W: 3x3=9

3. b: 3

4.V:2x3=6

5. c: 2

12+9+3+6+2=32

5. LSTM

LSTM（长短期记忆）是一种能够更好处理长距离依赖的 RNN 变体。 LSTM (Long Short-Term Memory) is a variant of RNN designed to better handle long-term dependencies.

引入记忆单元（Cell State），用于长期信息的存储； It introduces a memory cell (Cell State) to store long-term information;

使用三个门控机制： Uses three gate mechanisms:

遗忘门（Forget Gate）：决定哪些旧信息被丢弃； Forget Gate: decides what old information to discard;

输入门（Input Gate）：决定哪些新信息写入记忆； Input Gate: decides what new information to write into memory;

输出门（Output Gate）：决定输出什么； Output Gate: decides what to output;

能缓解传统 RNN 的梯度消失问题，适用于文本生成、机器翻译等任务； Helps alleviate the gradient vanishing problem in traditional RNNs and is used in text generation, translation, etc.;

计算结构更复杂，但性能更稳定。 More complex computationally but more stable in performance.

6. GRU

GRU（门控循环单元）是另一种简化版的 LSTM。 GRU (Gated Recurrent Unit) is a simplified version of LSTM.

合并了输入门与遗忘门，减少了计算复杂度； It merges the input and forget gates, reducing computational complexity;

核心组件： Core components:

重置门（Reset Gate）：控制如何结合新输入与过去记忆； Reset Gate: controls how to combine new input with past memory;

更新门（Update Gate）：控制保留多少过去的状态； Update Gate: controls how much past state to retain;

没有独立的 cell state，直接在 hidden state 上更新； No separate cell state; updates are made directly to the hidden state;

结构更简单，训练更快，适合小数据场景。 Simpler structure, faster training, suitable for small data scenarios.

十、text classification

1. One-hot Encoding

One-hot 编码是一种将词语或类别转换为向量的方式。 One-hot encoding is a method of converting words or categories into vectors.

每个 token 被表示为一个全 0 向量，只有其对应索引位置为 1； Each token is represented by an all-zero vector with only its index position set to 1;

如果词汇表大小为 N，则每个 token 表示为一个 N 维向量； If the vocabulary size is N, each token is represented by an N-dimensional vector;

示例：若“bird”是第 3 类，则表示为 [0, 0, 1, 0, 0, ...]。 Example: If 'bird' is the third class, it's represented as [0, 0, 1, 0, 0, ...].

2. Word Embedding

词嵌入是一种将单词映射到稠密向量空间的方法，使得相似语义的单词彼此接近。 Word embedding is a method of mapping words to dense vector space so that semantically similar words are close to each other.

每个词被编码为一个 M 维实数向量； Each word is encoded as an M-dimensional real-valued vector;

这些向量可以捕捉词的上下文和语义信息； These vectors capture the context and semantic meaning of words;

基于 distributional hypothesis：“上下文相似的词具有相似含义”。 Based on the distributional hypothesis: 'Words with similar contexts have similar meanings'.

3. Skip-gram

Skip-gram 是一种训练 word embedding 的方法，由 Word2Vec 提出： Skip-gram is a method for training word embeddings, proposed in Word2Vec:

输入一个目标词，模型预测其上下文中的词； Given a target word, the model predicts surrounding context words;

上下文通过一个固定大小的窗口采样； Context is sampled using a fixed-size window;

使用一个简单的 encoder-decoder 神经网络进行建模； A simple encoder-decoder neural network is used for modeling;

输出为词汇表上每个词的概率。 The output is a probability distribution over the vocabulary.

4. Demonstrate an example of using word embedding for text classification

使用词嵌入进行文本分类的示例： An example of using word embedding for text classification:

数据集：20 Newsgroups，包含 20,000 条文本，分属 20 个新闻主题； Dataset: 20 Newsgroups, with 20,000 documents across 20 news topics;

提取每个文档前 s 个词； Extract the first s words from each document;

每个词查表获得一个 50 维嵌入向量； Each word is mapped to a 50-dimensional embedding vector;

将这些向量输入到线性分类器； These vectors are input to a linear classifier;

输出为分类 ID，对应所属新闻组。 The output is a classification ID representing the news group.

5. Text Classification using CNN

使用 CNN 进行文本分类： Using CNN for text classification:

将文本表示为字符序列（如长度为 s）； Represent the text as a character sequence (e.g., of length s);

每个字符查表获得一个 16 维向量，形成一个 16×s 的矩阵； Each character is mapped to a 16-dimensional vector, forming a 16×s matrix;

使用 1D 卷积提取时间上下文特征； Use 1D convolution to extract temporal context features;

使用多个卷积块加池化层提取层次特征； Use multiple convolution blocks with pooling layers to extract hierarchical features;

最后通过全连接层 + Softmax 分类。 Finally classify with a fully connected layer and Softmax.

6. K-max Pooling

K-max pooling 是一种池化操作： K-max pooling is a type of pooling operation:

从长度为 n 的序列中选择最大值的前 k 个子序列； Select the top k largest values from a sequence of length n;

保留它们在原序列中的顺序； Preserve their order in the original sequence;

与 max pooling 只取一个最大值不同，k-max 可保留多个重要特征； Unlike max pooling that selects only one maximum, k-max retains multiple important features;

示例：输入序列 p = [3, 71, 9, 2, 1, 19, 27, 13, 6, 49]，k = 4 → 输出 q = [71, 19, 27, 49] Example: input sequence p = [3, 71, 9, 2, 1, 19, 27, 13, 6, 49], k = 4 → output q = [71, 19, 27, 49]

7. Cosine Similarity

余弦相似度用于衡量两个向量之间的夹角（相似度）： Cosine similarity measures the angle (similarity) between two vectors:

常用于文本向量之间的比较； Commonly used for comparing text vectors;

余弦值越接近 1，表示越相似； The closer the cosine value is to 1, the more similar the vectors are;

90° → 相似度为 0（完全不相似）； 90° → similarity is 0 (completely dissimilar);

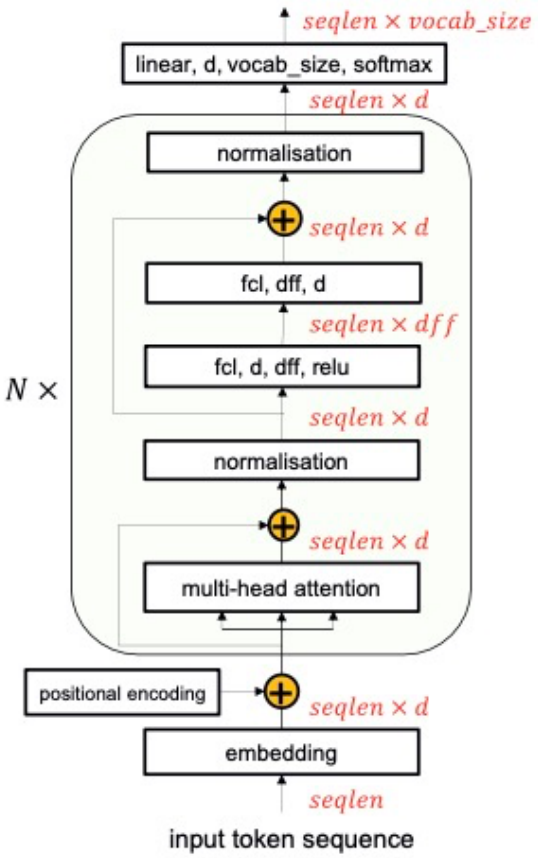
0° → 相似度为 1（完全相似）； 0° → similarity is 1 (perfectly similar);

应用场景：文本聚类、检索、推荐系统。 Use cases: text clustering, retrieval, recommendation systems;

比如判断“pound 与 England”与“yen 与 Japan”的关系相似性。 For example, measuring relational similarity between 'pound and England' vs 'yen and Japan'.

十一、transformers

GPT Transformer



1 输入编码（Input Encoding）

原始文本要先被“分词”，称为 tokenize。 Raw text must first be 'tokenized'.

比如：“The cat sat” → 变成：[“The”, “Ġcat”, “Ġsat”] For example: 'The cat sat' → becomes: ['The', 'Ġcat', 'Ġsat']

这里的 “Ġ” 表示空格，说明用了 Byte Pair Encoding (BPE) 的编码方法。 The 'Ġ' indicates a space, showing the use of Byte Pair Encoding (BPE).

每个 token 被映射为一个向量（embedding），组成一个序列矩阵。 Each token is mapped to a vector (embedding), forming a sequence matrix.

例如，GPT-3 的输入长度可以达到 2048 个 token，每个 token 是一个固定维度的向量（如 768、1024 等）。 For example, GPT-3 can accept up to 2048 tokens, each as a fixed-dimensional vector (e.g., 768, 1024).

2.注意力机制（Multi-head Attention）

每个 token 的表示，不只由它自身决定，而是通过“注意力机制”从其他所有 token 处获取信息。 Each token's representation is not determined solely by itself, but gathers information from all other tokens through 'attention mechanism'.

输入会被投影成三个向量：Query, Key, Value。 Inputs are projected into three vectors: Query, Key, and Value.

使用这些向量计算 token 之间的权重（相关性）。 Use these vectors to compute weights (correlations) between tokens.

多个注意力头（multi-head）可捕捉多种不同的语义关系。 Multiple attention heads capture various semantic relationships.

这就是所谓的 “self-attention” 机制，Transformer 的核心组件。 This is known as the 'self-attention' mechanism, the core of the Transformer.

引入了三个重要向量:

Query (Q) 我是谁 我想要什么

Key (K) 你是谁 你能提供什么

Value (v) 你给我的内容具体是什么

每个词会产生自己的 Q、K、V;

然后我们用 Q 和所有 K 做点积，得到“匹配程度"(注意力分数);

最后，用这些分数对所有 V 做加权平均，得出最终的注意力输出。

3 归一化（Normalisation）

在每个子模块后使用 LayerNorm 进行层归一化，以稳定训练。 Layer normalization (LayerNorm) is applied after each sub-module to stabilize training.

类似于标准化数据，帮助梯度传播更顺利。 Similar to data standardization, it helps gradients propagate more smoothly.

4 位置编码（Positional Encoding）

Transformer 不像 RNN，有天然的“时间顺序”。 Unlike RNNs, Transformers lack a natural 'temporal order'.

所以我们需要告诉模型：token 的位置是多少。 So we need to inform the model about each token's position.

使用一种叫“正弦+余弦函数”的方法，把每个位置信息编码为一个向量； A method called 'sine and cosine functions' encodes positional information into a vector;

GPT-3 中这个位置编码是可学习的向量。 In GPT-3, this positional encoding is a learnable vector.

这样每个 token 的 embedding = 词向量 + 位置信息。 Thus, each token's embedding = word vector + positional information.

5 输出层（Output Layer）

Transformer 最后一层的输出仍然是 seqlen × d 的矩阵； The final Transformer layer outputs a seqlen × d matrix;

我们用一个线性层将其投影为词表大小 vocab_size； We use a linear layer to project it to vocab_size;

再使用 softmax，得到每个位置上可能输出哪个 token 的概率。 Then softmax gives the probability of each possible output token per position.

6 模型训练（Training）

训练目标：让模型能预测下一个词。 Training goal: Enable the model to predict the next word.

步骤如下： Steps are as follows:

1. 给模型输入一个 token 序列，如 “I like deep” 1. Provide the model a token sequence, e.g., 'I like deep'

2. 目标是预测：“like deep learning” 2. Target is to predict: 'like deep learning'

3. 损失函数使用的是 negative log-likelihood（负对数似然） 3. Loss function used is negative log-likelihood

目的：预测 “learning” 的概率越高，损失越小 Goal: The higher the probability of 'learning', the lower the loss.

7 文本生成（Evaluation）

训练好之后，我们就可以用它来生成新文本了。 Once trained, we can use it to generate new text.

步骤： Steps:

1. 给一个初始文本，如：“Joaquin Phoenix pledged” 1. Provide an initial text like: 'Joaquin Phoenix pledged'

2. 填满 2048 token 的 context window（如果不足就 pad） 2. Fill up the 2048-token context window (pad if needed)

3. 用 Transformer 输出最后一个 token 的概率分布； 3. Use the Transformer to output the probability distribution of the last token;

4. 从中采样一个 token，接到原始输入后继续： 4. Sample one token and append it to the input to continue:

重复喂入模型，预测下一个 token； Repeat feeding into the model to predict the next token;

重复，直到达到所需长度。 Repeat until the desired length is reached.

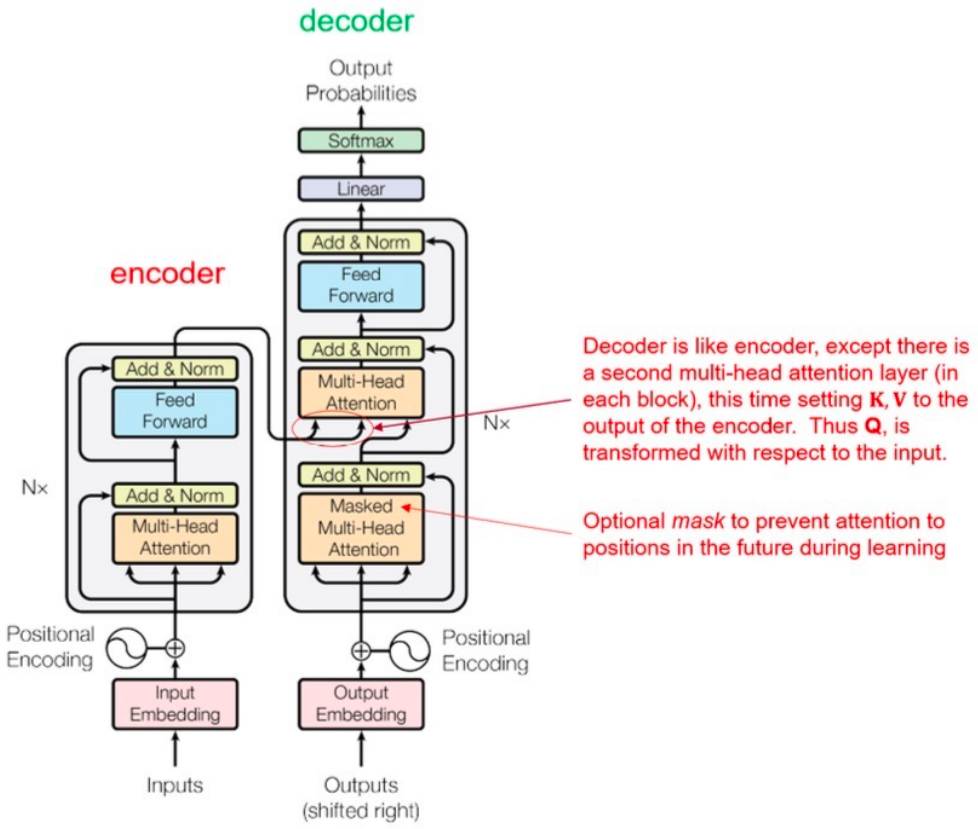
注意： Note:

每次生成新的 token，都重新运行 Transformer； Each time a token is generated, the Transformer is rerun;

而不像 RNN 可以一步接一步往前推。 Unlike RNNs which can proceed step-by-step.

GPT 是“自回归”生成器：已有前面的内容，预测下一个词。 GPT is an autoregressive generator: it predicts the next word based on previous content.

Full Transformer



1. Encoder

Encoder 做什么

将整个输入句子编码成一个上下文相关的向量序列;每一个输出向量都"知道"这个词在整个句子中的含义

特点:

Encoder 是多个 Transformer Block 的堆养;

每个 block 包括:多头注意力+前馈网络+LayerNorm;输入是 token embedding + position embedding ;输出 shape 和输入一样，只是"更有语义了"

2. Decoder

Decoder 的目标是逐词生成输出句子。Docoder 可以同时看我已经生成了什么，原文句子里说了什么。

Docoder 如何生成句子:

这个过程是自回归的(auto-regressive)，也就是一个词一个词往下生 1.输入原文:"The cat is on the mat"

2.Decoder 输入 <start>，输出概率最高的 token:比如 "Le"3.将 <start> Le 作为新的 Decoder 输入，再预测下一个 token →"chat"4.变成 <start> Le chat ，再生成下一个 token...5.一直重复，直到模型输出 <end>为止

这个过程就是 GPT、翻译模型、写诗机器人等的工作方式。

把 Encoder 想象成看完整篇文章的读者，把关键内容记下来;把 Decoder 想象成逐字写文章的学生，边写边参考"读者的摘要"。Encoder 把输入的含义浓缩了; Decoder 每次写一个字都"参考"这个摘要；而且还知道"我前面已经写了什么"。

十二、relational data

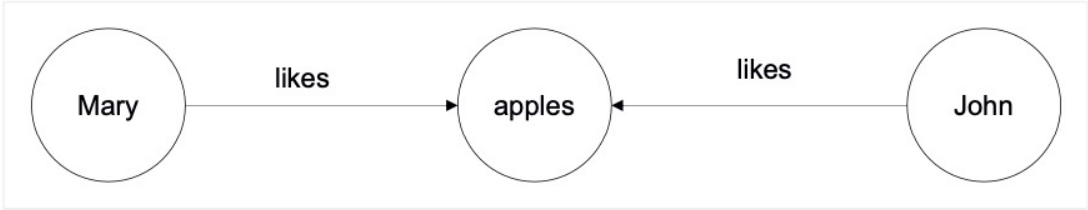


Figure 1. A graph of three nodes representing two binary relationships.

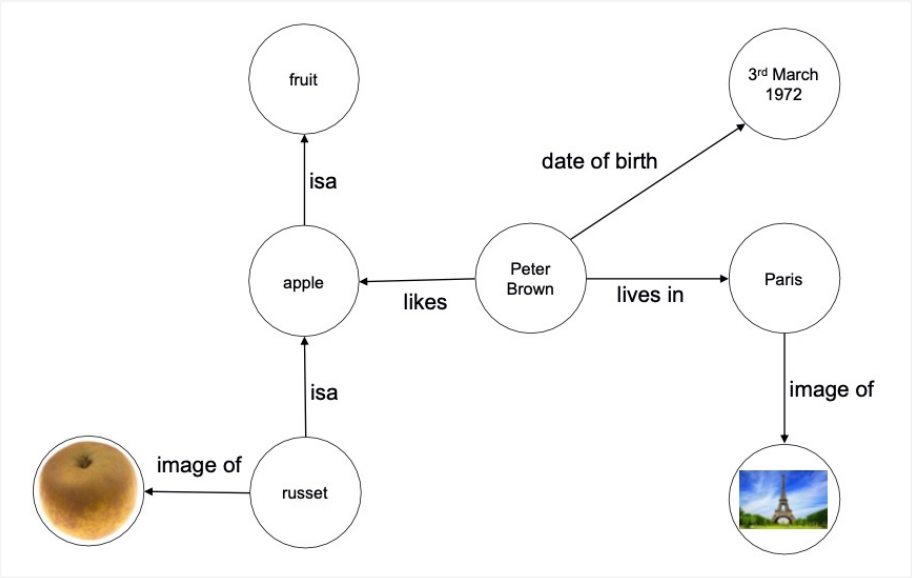


Figure 2. A fragment of knowledge graph representing a collection of facts.

十三、GNN graph neural networks

GNN 流程
1.每条边根据（它自己，begin 节点，end 节点，全局数据）更新它自己的数据
2.汇总每个节点的入度边的数据
3.每个节点根据（汇总数据，它自己的数据，全局数据）更新自己的数据
4.全局数据汇总所有边数据
5.全局数据汇总所有节点数据
6.全局数据更新自己的数据

十四、Autoencoders

1. Autoencoder

自动编码器是一种无监督学习方法，能在不需要标签的情况下学习数据的表示。 Autoencoders are an unsupervised learning method that can learn representations of data without labels.

它通过最小化输入与输出之间的重构误差来学习。 It learns by minimizing the reconstruction loss between input and output.

学习包括两个函数：编码器将输入压缩为低维向量，解码器从向量重建输入。 It learns two functions: an encoder that compresses the input and a decoder that reconstructs it.

编码表示是低维向量，称为“代码”或“潜在向量”。 The encoded representation is a low-dimensional vector, called a 'code' or 'latent vector'.

可以训练单隐层，也可堆叠多隐层形成深度自动编码器。 It can be trained with a single hidden layer, or stacked into deep autoencoders with multiple hidden layers.

也可用于生成与训练数据相似的新数据。 They can also generate new data similar to the training set.

2. components of Autoencoder

典型结构包括三个部分：编码器、潜在向量（瓶颈层）和解码器。 A typical structure includes three parts: encoder, latent vector (bottleneck), and decoder.

编码器：将输入映射为潜在空间的向量。 Encoder: maps input to a vector in latent space.

解码器：从潜在向量重构输入。 Decoder: reconstructs the input from the latent vector.

瓶颈层包含压缩后的信息。 The bottleneck layer contains compressed information.

训练时最小化重构损失（如 L2 距离）。 Training minimizes reconstruction loss (e.g., L2 distance).

3. Applications of Autoencoders

高维数据可视化（如映射到 2D） High-dimensional data visualization (e.g., mapping to 2D)

数据压缩 Data compression

图像去噪 Image denoising

图像补全 Image inpainting

图像检索 Image retrieval

4. Types of Autoencoders

欠完备自动编码器（Undercomplete AE） Undercomplete autoencoder

稀疏自动编码器（Sparse AE） Sparse autoencoder

收缩自动编码器（Contractive AE） Contractive autoencoder

去噪自动编码器（Denoising AE） Denoising autoencoder

变分自动编码器（Variational AE） Variational autoencoder

5. Denoising Autoencoder

其目标是从带噪声的输入中恢复原始干净的输入。 The goal is to recover clean input from noisy input.

输入被加入高斯噪声或其他形式的干扰。 Noise such as Gaussian noise is added to the input.

训练后可以有效还原原始图像。 After training, it can effectively restore the original image.

6. Variational Autoencoder (VAE)

VAE 是一种概率生成模型。 VAE is a probabilistic generative model.

潜在向量 z 不是确定值，而是从某个分布中采样（如正态分布）。 Latent vector z is sampled from a distribution (e.g., normal distribution) rather than being deterministic.

使用均值和标准差描述分布。 The distribution is described using mean and standard deviation.

VAE 在损失函数中引入正则项以逼近标准正态分布。 VAE includes a regularization term in the loss to approximate a standard normal distribution.

7. Loss function in VAE

VAE 的损失由两部分组成： The VAE loss consists of two parts:

1）重构损失：确保输出图像逼近输入。 1) Reconstruction loss: ensures the output resembles the input.

2）KL 散度：测量采样分布与标准正态分布之间的差距。 2) KL divergence: measures the difference between sampled and standard normal distribution.

8. Reparameterisation Trick

VAE 中采样操作不可导，无法进行反向传播。 Sampling in VAE is non-differentiable, which prevents backpropagation.

重参数化技巧将采样写作：均值 + 噪声 × 方差，使其可导。 The reparameterization trick rewrites sampling as: mean + noise × std, making it differentiable.

该方法使梯度能够传递，训练可行。 This allows gradients to flow and makes training feasible.

9. Applications of VAE

图像生成 Image generation

图像修复（inpainting） Image inpainting

脑部病灶检测 Brain lesion detection

内窥图像分割 Endoscopic image segmentation

十五、Diffusion models

Diffusion Models - Bilingual Summary

1. Diffusion Model

Diffusion Model（扩散模型）是一类概率生成模型，其核心思想是： Diffusion Models are a class of probabilistic generative models, whose core idea is:

前向过程：逐步将数据加入高斯噪声，直到接近纯噪声； Forward process: gradually add Gaussian noise to the data until it becomes near-pure noise;

反向过程：从纯噪声开始，通过去噪步骤逐渐恢复原始图像。 Reverse process: starts from pure noise and denoises step by step to recover the original image.

这种结构是一种马尔可夫链，每一步都基于前一步状态。 This structure forms a Markov Chain, where each step is based on the previous state.

2. Variational Diffusion Model (VDM)

变分扩散模型是扩散模型的一种形式，具有以下特点： Variational Diffusion Model is a type of diffusion model with the following characteristics:

可以看作是 Markovian 层次变分自动编码器（H-VAE）的一种特殊情况； It can be regarded as a special case of Markovian hierarchical variational autoencoder (H-VAE);

每一步的潜在编码器是预定义的线性高斯模型； Each latent encoder at each step is a predefined linear Gaussian model;

潜变量维度与数据维度相同； The dimension of the latent variable is the same as the data;

每一步进行高斯扰动，反向过程则是从噪声一步步“去噪”恢复数据。 Each step involves Gaussian corruption, and the reverse process gradually denoises to recover the data.

3. Variational Diffusion Model: Optimization

VDM 的优化目标是： The optimization objective of VDM is:

保证每一步的潜变量都匹配对应的高斯扰动； Ensure that the latent variable at each step matches the corresponding Gaussian corruption;

前向过程是固定的，只需学习反向过程； The forward process is fixed, and only the reverse process needs to be learned;

优化目标是让模型生成的反向分布与真实后验分布尽可能接近； The goal is to make the learned reverse distribution as close as possible to the true posterior;

最终潜变量服从标准高斯分布，可以用来采样生成图像。 The final latent variable follows a standard Gaussian distribution, which can be used to sample images.

4. Latent Diffusion Model (LDM)

LDM（潜空间扩散模型）是将扩散过程放在 VAE 编码后的低维潜空间中进行： LDM (Latent Diffusion Model) performs the diffusion process in the low-dimensional latent space encoded by a VAE:

架构包括： Its architecture includes:

一个 VAE（用于从图像编码到潜空间） A VAE (used to encode images into latent space)

一个 U-Net（用于扩散过程中的去噪） A U-Net (for denoising in the diffusion process)

一个文本编码器（如 CLIP，用于条件控制） A text encoder (e.g., CLIP, for conditional control)

优点： Advantages:

降低了扩散过程的计算复杂度； Reduces computational cost of the diffusion process;

保留语义特征而去除高频细节； Preserves semantic features while removing high-frequency details;

可用于高分辨率图像生成； Enables high-resolution image generation;

模型在低维空间生成，再通过解码器还原成原图。 The model generates in low-dimensional space, then reconstructs to image space via a decoder.

5. What are the Applications of Diffusion Model

图像超分辨率（Super-resolution） Image super-resolution

图像修复（Inpainting） Image inpainting

文本生成图像（Text-to-Image Synthesis） Text-to-image synthesis

图像条件生成（Image-conditioned Image Generation） Image-conditioned image generation

十六、GAN and image to image translation

1. Generative Adversarial Networks (GAN)

GAN 是一种生成模型，由两个神经网络组成：生成器和判别器，它们在一个对抗性设置下进行训练。 GAN is a generative model consisting of two neural networks: a generator and a discriminator, trained in an adversarial setting.

生成器的目标是从随机噪声中生成“逼真”的假图像。 The generator aims to produce 'realistic' fake images from random noise.

判别器的目标是区分图像是真实的还是伪造的。 The discriminator aims to distinguish whether an image is real or fake.

这是一个零和博弈，两个模型相互对抗，最终生成器能“骗过”判别器，生成出高度真实的图像。 It's a zero-sum game where the models compete until the generator can 'fool' the discriminator with highly realistic images.

2. How does GAN work

GAN 的训练是一个交替优化过程： GAN training is an alternating optimization process:

给定真实图像和生成器生成的伪造图像，判别器输出每张图像是真实的概率。 Given real images and generator-generated fakes, the discriminator outputs the probability of each image being real.

训练目标是最大化真实图像的 log 概率，最小化伪造图像的 log(1 − 概率)。 The goal is to maximize log probability of real images and minimize log(1 - probability) for fake ones.

生成器则被训练去最大化伪图被判定为真实的概率。 The generator is trained to maximize the probability that the fake images are considered real.

3. input and output of GAN

输入（生成器）：一个随机向量，例如从正态分布或均匀分布中采样。 Input (to the generator): a random vector sampled from a normal or uniform distribution.

输出：一张“伪造”的图像，例如 28×28 的手写数字或 256×256 的自然图像。 Output: a 'fake' image, such as a 28×28 handwritten digit or a 256×256 natural image.

4. How to train GAN

训练判别器： Train the discriminator:

从训练集中采样 m 张真实图像。 Sample m real images from the dataset.

用生成器生成 m 张伪图。 Generate m fake images using the generator.

使用交叉熵损失函数最大化真实图和伪图的分类准确性。 Use cross-entropy loss to maximize classification accuracy of real and fake images.

训练生成器： Train the generator:

固定判别器参数，更新生成器以提高伪图被判定为真实的概率。 Fix the discriminator and update the generator to increase the probability that fake images are classified as real.

这两个过程交替进行，形成联动训练。 These two processes alternate, forming a linked training procedure.

5. image-to-image translation

图像到图像的转换是指将一张图像从一个域转换为另一个。 Image-to-image translation refers to converting an image from one domain to another.

如黑白图 → 彩色图，夏天风景 → 冬天风景，边缘图 → 实物图。 For example: grayscale → color, summer scene → winter scene, edge map → real object.

这种任务不一定需要成对的训练数据。 This task does not necessarily require paired training data.

6. CycleGAN

CycleGAN 是一种专为无配对数据设计的图像翻译框架。 CycleGAN is an image translation framework designed for unpaired data.

它使用两个生成器（X→Y 和 Y→X）和两个判别器（分别处理 X 和 Y）进行域间转换。 It uses two generators (X→Y and Y→X) and two discriminators (for X and Y) for domain translation.

通过循环一致性损失，强制图像从 X → Y → X’ 后 X’ ≈ X。 Cycle-consistency loss forces X → Y → X’ such that X’ ≈ X.

7. The Reason CycleGAN can be trained without pair images

CycleGAN 利用了循环一致性损失。 CycleGAN uses cycle-consistency loss.

将图像 X 转换为 Y，然后再转换回 X’，要求 X’ ≈ X。 Image X is translated to Y, then back to X’, and it is required that X’ ≈ X.

同样地，Y → X → Y’ ≈ Y。 Similarly, Y → X → Y’ ≈ Y.

这个约束使得模型不需要成对图像即可训练。 This constraint enables training without paired images.