浅析CNN算法及其人工智能应用

结合 Alex,Wang 的研究 与 AI图像检测

侯东杨

北京林业大学 理学院

2025 年 6 月 5 日

概述

卷积神经网络算法

Convolutional Neural Network,简称CNN,是一种深度学习算法,专门用于处理具有网格状拓扑结构数据.

CNN能够有效提取数据的局部特征并保持空间不变性,在计算机视觉任务(如图像分类、目标检测、图像分割等)中表现尤为出色,因此其在当前的生成式人工智能识别领域具有非常广泛的应用前景.

research highlights

DOI:10.1145/3065386

ImageNet Classification with Deep Convolutional Neural Networks

By Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0%, respectively, which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully connected layers we employed a recently developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1. PROLOGUE

Four years ago, a paper by Yann LeCun and his collaborators was rejected by the leading computer vision conference on the grounds that it used neural networks and therefore provided no insight into how to design a vision system. At the time, most computer vision researchers believed that a vision system needed to be carefully hand-designed using a detailed understanding of the nature of the task. They assumed that the task of classifying objects in natural images would never be solved by simply presenting examples of images and the names of the objects they contained to a neural network that acquired all of its knowledge from this training data.

What many in the vision research community failed to appreciate was that methods that require careful hand-engineering by a programmer who understands the domain do not scale as well as methods that replace the programmer with a powerful general-purpose learning procedure. With enough computation and enough data, learning beats programming for complicated tasks that require the integration of many different, noisy cues.

Four years ago, while we were at the University of Toronto, our deep neural network called SuperVision almost halved the error rate for recognizing objects in natural images and triggered an overdue paradigm shift in computer vision. Figure 4 shows some examples of what SuperVision can do.

SuperVision evolved from the multilayer neural networks that were widely investigated in the 1980s. These networks used multiple layers of feature detectors that were all learned from the training data. Neuroscientists and psychologists had hypothesized that a hierarchy of such feature detectors would provide a robust way to recognize objects but they had no idea how such a hierarchy could be learned. There was great excitement in the 1980s because several different research groups discovered that multiple layers of feature detectors could be trained efficiently using a relatively straight-forward algorithm called backpropagation[9,10,25,31] to compute, for each image, how the classification performance of the whole network depended on the value of the weight on each connection.

Backpropagation worked well for a variety of tasks, but in the 1980s it did not live up to the very high expectations of its advocates. In particular, it proved to be very difficult to learn networks with many layers and these were precisely the networks that should have given the most impressive results. Many researchers concluded, incorrectly, that learning a deep neural network from random initial weights was just too difficult. Twenty years later, we know what went wrong: for deep neural networks to shine, they needed far more labeled data and hugely more computation.

2. INTRODUCTION

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small—on the order of tens of thousands of images (e.g., NORB,[19] Caltech-101/256,[8,10] and CIFAR-10/100[18]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance.[3] But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Ref.[23]), but it has only recently become possible to collect labeled datasets with millions of

The original version of this paper was published in the *Proceedings of the 25th International Conference on Neural Information Processing Systems* (Lake Tahoe, NV, Dec. 2012), 1097–1105.

84 COMMUNICATIONS OF THE ACM | JUNE 2017 | VOL. 60 | NO. 6

CNN-generated images are surprisingly easy to spot... for now

Sheng-Yu Wang[1] Oliver Wang[2] Richard Zhang[2] Andrew Owens[1,3] Alexei A. Efros[1]

UC Berkeley[1] Adobe Research[2] University of Michigan[3]

ProGAN [21] StyleGAN [22] BigGAN [9] CycleGAN [54] StarGAN [12] GauGAN [36] CRN [11] IMLE [26] SITD [10] Super-res. [15] DeepFakes [39]

Figure 1: **Are CNN-generated images hard to distinguish from real images?** We show that a classifier trained to detect images generated by only one CNN (ProGAN, far left) can detect those generated by many other models (remaining columns). Our code and models are available at https://peterwang512.github.io/CNNDetection/.

Abstract

In this work we ask whether it is possible to create a "universal" detector for telling apart real images from these generated by a CNN, regardless of architecture or dataset used. To test this, we collect a dataset consisting of fake images generated by 11 different CNN-based image generator models, chosen to span the space of commonly used architectures today (ProGAN, StyleGAN, BigGAN, CycleGAN, StarGAN, GauGAN, DeepFakes, cascaded refinement networks, implicit maximum likelihood estimation, second-order attention super-resolution, seeing-in-the-dark). We demonstrate that, with careful pre- and post-processing and data augmentation, a standard image classifier trained on only one *specific CNN generator (ProGAN) is able to generalize surprisingly well to unseen architectures, datasets, and training methods (including the just released StyleGAN2 [23]). Our findings suggest the intriguing possibility that today's CNN-generated images share some common systematic flaws, preventing them from achieving realistic image synthesis.*

1. Introduction

Recent rapid advances in deep image synthesis techniques, such as Generative Adversarial Networks (GANs), have generated a huge amount of public interest and concern, as people worry that we are entering a world where it will be impossible to tell which images are real and which are fake [16]. This issue has started to play a significant role in global politics: in one case a video of the president of Gabon that was claimed by opposition to be fake was one factor leading to a failed coup d'etat*. Much of this concern has been directed at specific manipulation techniques, such as "deepfake"-style face replacement [3], and photo-realistic synthetic humans [22]. However, these methods represent only two *instances* of a broader set of techniques: image synthesis via convolutional neural networks (CNNs). Our goal in this work is to find a general image forensics approach for detecting CNN-generated imagery.

Detecting whether an image was generated by a specific synthesis technique is relatively straightforward—just train a classifier on a dataset consisting of real images and images synthesized by the technique in question. However, such an approach will likely be tied to the dataset used in image generation (*e.g.* faces), and, due to dataset bias [41], might not generalize when tested on new data (*e.g.* cars). Even worse, the technique-specific detector is likely to soon become ineffective as generation methods evolve and the technique it was trained on becomes obsolete.

It is natural, therefore, to ask whether today's CNN-generated images contain common artifacts, *e.g.*, some kind of detectable CNN *fingerprints*, that would allow a classifier to generalize to an entire family of generation methods, rather than a single one. Unfortunately, prior work has reported generalization to be a significant problem for

*https://www.motherjones.com/politics/2019/03/deepfake-gabon-ali-bongo/

1

概述

research highlights

DOI:10.1145/3065386

ImageNet Classification with Deep Convolutional Neural Networks

By Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0%, respectively, which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully connected layers we employed a recently developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1. PROLOGUE

Four years ago, a paper by Yann LeCun and his collaborators was rejected by the leading computer vision conference on the grounds that it used neural networks and therefore provided no insight into how to design a vision system. At the time, most computer vision researchers believed that a vision system needed to be carefully hand-designed using a detailed understanding of the nature of the task. They assumed that the task of classifying objects in natural images would never be solved by simply presenting examples of images and the names of the objects they contained to a neural network that acquired all of its knowledge from this training data.

What many in the vision research community failed to appreciate was that methods that require careful hand-engineering by a programmer who understands the domain do not scale as well as methods that replace the programmer with a powerful general-purpose learning procedure. With enough computation and enough data, learning beats programming for complicated tasks that require the integration of many different, noisy cues.

Four years ago, while we were at the University of Toronto, our deep neural network called SuperVision almost halved the error rate for recognizing objects in natural images and triggered an overdue paradigm shift in computer vision. Figure 4 shows some examples of what SuperVision can do.

SuperVision evolved from the multilayer neural networks that were widely investigated in the 1980s. These networks used multiple layers of feature detectors that were all learned from the training data. Neuroscientists and psychologists had hypothesized that a hierarchy of such feature detectors would provide a robust way to recognize objects but they had no idea how such a hierarchy could be learned. There was great excitement in the 1980s because several different research groups discovered that multiple layers of feature detectors could be trained efficiently using a relatively straight-forward algorithm called backpropagation[18, 22, 27, 33] to compute, for each image, how the classification performance of the whole network depended on the value of the weight on each connection.

Backpropagation worked well for a variety of tasks, but in the 1980s it did not live up to the very high expectations of its advocates. In particular, it proved to be very difficult to learn networks with many layers and these were precisely the networks that should have given the most impressive results. Many researchers concluded, incorrectly, that learning a deep neural network from random initial weights was just too difficult. Twenty years later, we know what went wrong: for deep neural networks to shine, they needed far more labeled data and hugely more computation.

2. INTRODUCTION

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small—on the order of tens of thousands of images (e.g., NORB,[19] Caltech-101/256,[8, 10] and CIFAR-10/100[14]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance.[5] But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Ref.[25]), but it has only recently become possible to collect labeled datasets with millions of

The original version of this paper was published in the *Proceedings of the 25th International Conference on Neural Information Processing Systems* (Lake Tahoe, NV, Dec. 2012), 1097–1105.

(a) Alex Krizhevsky 2012

CNN-generated images are surprisingly easy to spot... for now

Sheng-Yu Wang[1] Oliver Wang[2] Richard Zhang[2] Andrew Owens[1,3] Alexei A. Efros[1]
UC Berkeley[1] Adobe Research[2] University of Michigan[3]

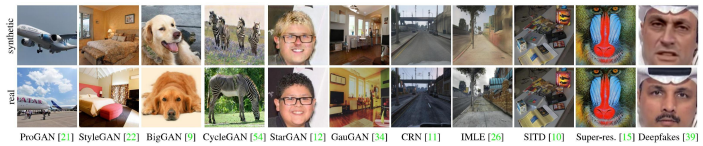


Figure 1: **Are CNN-generated images hard to distinguish from real images?** We show that a classifier trained to detect images generated by only one CNN (ProGAN, far left) can detect those generated by many other models (remaining columns). Our code and models are available at https://peterwang512.github.io/CNNDetection/.

Abstract

In this work we ask whether it is possible to create a "universal" detector for telling apart real images from these generated by a CNN, regardless of architecture or dataset used. To test this, we collect a dataset consisting of fake images generated by 11 different CNN-based image generator models, chosen to span the space of commonly used architectures today (ProGAN, StyleGAN, BigGAN, CycleGAN, StarGAN, GauGAN, DeepFakes, cascaded refinement networks, implicit maximum likelihood estimation, second-order attention super-resolution, seeing-in-the-dark). We demonstrate that, with careful pre- and post-processing and data augmentation, a standard image classifier trained on only one *specific CNN generator (ProGAN) is able to generalize surprisingly well to unseen architectures, datasets, and training methods (including the just released StyleGAN2 [23]). Our findings suggest the intriguing possibility that today's CNN-generated images share some common systematic flaws, preventing them from achieving realistic image synthesis.*

1. Introduction

Recent rapid advances in deep image synthesis techniques, such as Generative Adversarial Networks (GANs), have generated a huge amount of public interest and concern, as people worry that we are entering a world where it will be impossible to tell which images are real and which are fake [16]. This issue has started to play a significant role in global politics; in one case a video of the president of Gabon that was claimed by opposition to be fake was one factor leading to a failed coup d'etat*. Much of this concern has been directed at specific manipulation techniques, such as "deepfake"-style face replacement [3], and photo-realistic synthetic humans [22]. However, these methods represent only two *instances* of a broader set of techniques: image synthesis via convolutional neural networks (CNNs). Our goal in this work is to find a general image forensics approach for detecting CNN-generated imagery.

Detecting whether an image was generated by a specific synthesis technique is relatively straightforward — just train a classifier on a dataset consisting of real images and images synthesized by the technique in question. However, such an approach will likely be tied to the dataset used in image generation (*e.g.* faces), and, due to dataset bias [41], might not generalize when tested on new data (*e.g.* cars). Even worse, the technique-specific detector is likely to soon become ineffective as generation methods evolve and the technique it was trained on becomes obsolete.

It is natural, therefore, to ask whether today's CNN-generated images contain common artifacts, *e.g.*, some kind of detectable CNN *fingerprints*, that would allow a classifier to generalize to an entire family of generation methods, rather than a single one. Unfortunately, prior work has reported generalization to be a significant problem for

*https://www.motherjones.com/politics/2019/03/deepfake-gabon-ali-bongo/

(b) Sheng-Yu Wang 2020

目录

卷积神经网络

Convolutional Neural Network,简称CNN,是一种深度学习算法,专门用于处理具有网格状拓扑结构数据.

- **CNN的核心思想**
  CNN的设计灵感来源于人脑视觉系统,特别是视觉皮层中神经元对局部感受的响应.其模拟人脑视觉系统,提取局部特征,通过层次结构实现特征的抽象与整合.
- **CNN的算法流程**
  CNN通过**卷积层**提取局部特征,结合**池化层**降低维度,**全连接层**整合信息,用于分类、检测等任务,同时可引入**正则化层**防止过拟合.其利用参数共享和空间层次结构,有效减少计算量,提升对视觉数据的处理能力.

§2.1核心组件:卷积层

卷积层

应用指定大小的**卷积核**在经过预处理(归一化、裁剪尺寸、添加额外像素等)的输入数据上滑动,提取对应数据的局部特征(边缘、纹理、形状等),形成**特征图**,同时保留空间结构信息.

- **卷积核**
  - 一般为定义小型矩阵(3阶或5阶方阵),在输入数据上滑动
  - 每个卷积核学习特定特征(如水平边缘、垂直边缘)
  - 每次的特定特征将参与形成本次卷积操作的**特征图**
- **特征图**
  - 卷积核与局部区域点积运算,生成特征图
  - 特征图形成公式:$(f*g)(i,j)=\sum_{m,n} f(m,n)g(i-m,j-n)$
  - 特征图的长宽尺寸:$W_{out}=[\frac{W_{in}-K+2P}{S}]+1$,深度(通道数)即为卷积核的数量
- **参数共享优势**
  - 同一卷积核应用于所有区域,减少参数量,提高计算效率
  - 每次特征图均由相同卷积核生成,增加输出结果的鲁棒性

§2.1核心组件:卷积层

卷积层

应用指定大小的**卷积核**在经过预处理(归一化、裁剪尺寸、添加额外像素等)的输入数据上滑动,提取对应数据的局部特征(边缘、纹理、形状等),形成**特征图**,同时保留空间结构信息.

- **数据设定**
  假设输入为一个单通道图像(对应为4阶方阵A),选取二阶方阵卷积核B,如下所示.

$$A=\begin{pmatrix}1&2&3&4\\5&6&7&8\\9&10&11&12\\13&14&15&16\end{pmatrix}\quad B=\begin{pmatrix}1&0\\0&-1\end{pmatrix}\quad C=\begin{pmatrix}5&5&5\\5&5&5\\5&5&5\end{pmatrix}$$

- **卷积操作**
  - 步长$S=1$,无填充,即$P=0$
  - 在特征图对应矩阵C位置$(1,1)$处,$(f*g)(1,1)=1\cdot 6+(-1)\cdot 1=5$
  - 滑动卷积核,计算所有位置,生成特征图,其对应矩阵即为C如上所示

§2.2核心组件:池化层

池化层

池化层通过在卷积层生成的特征图上应用指定大小的**池化窗口**进行下采样,**降低特征图维度**,保留关键特征信息,实现降低计算复杂度和增强模型的空间不变性的目标.

- **池化窗口**
  - 一般为小型矩阵(常取2阶、3阶方阵)在**特征图**上滑动
  - 常见池化方式:最大池化(取窗口内最大值)、平均池化(取窗口内均值)
  - 池化窗口通过步幅控制滑动距离,决定池化层输出尺寸
- **特征图降维**
  - 池化操作对特征图的局部区域进行聚合,生成更小的特征图
  - 输出降维特征图尺寸公式与卷积层相同,保留相同的算法
  - 通道数保持不变,每个输入特征图生成一个输出特征图
- **池化优势**
  - 减少特征图尺寸,进一步降低计算量和参数量,防止过拟合
  - 增强模型对位置变化的鲁棒性(平移不变性),进一步提高泛化能力

§2.2核心组件:池化层

池化层

池化层通过在卷积层生成的特征图上应用指定大小的**池化窗口**进行下采样,**降低特征图维度**,保留关键特征信息,实现降低计算复杂度和增强模型的空间不变性的目标.

- **数据设定**
  假设输入为一个单通道图像(对应为4阶方阵A),如下所示.

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \quad C = \begin{pmatrix} 6 & 8 \\ 14 & 16 \end{pmatrix}$$

- **池化窗口**
  使用一个2阶方阵的最大池化窗口,步长$S = 2$,无填充,即$P = 0$.
- **池化操作**
  - 左上角:$\max\{1\ 2\ 5\ 6\} = 6$,其余位置采取相同算法.
  - 最终得到池化层降低维度后的特征图,其对应矩阵即为C如上所示

§2.3核心组件:全连接层

全连接层

全连接层通过将卷积层或池化层生成的特征图展平为一维向量,并与权重矩阵进行全连接运算,整合全局特征,用于分类、回归等任务.

- **全连接操作**
  - 将输入特征图展平为一维向量x
  - 通过权重矩阵W和偏置b,计算输出:$y = Wx + b$
  - 通常配合激活函数,如Softmax用于分类,ReLU用于隐藏层
- **输出特征**
  - 输出为固定大小的向量,长度通常对应任务目标,如分类任务的类别数
  - 整合卷积层和池化层提取的局部特征,形成全局表示.
  - 若输入向量为$x \in \mathbb{R}^N$,权重矩阵$W \in \mathbb{R}^{M \times N}$则输出$y \in \mathbb{R}^M$
- **全连接优势**
  - 综合全局信息,适合多类别分类等高层次决策任务
  - 可与正则化结合,减少过拟合风险

§2.3核心组件:全连接层

全连接层

全连接层通过将卷积层或池化层生成的特征图展平为一维向量,并与权重矩阵进行全连接运算,整合全局特征,用于分类、回归等任务.

- **数据设定**
  假设输入为池化层的两个单通道特征图,其对应矩阵为2阶方阵A, B

$$A = \begin{pmatrix} 6 & 8 \\ 14 & 16 \end{pmatrix} B = \begin{pmatrix} 3 & 5 \\ 7 & 9 \end{pmatrix}$$

  展平为向量$x = (6, 8, 14, 16, 3, 5, 7, 9)^T$
- **全连接操作**
  使用权重矩阵$W \in \mathbb{R}^{2\times 8}$和偏置$b \in \mathbb{R}^2$目标为二分类任务:

$$W = \begin{pmatrix} 0.1 & 0.2 & 0.1 & 0.3 & 0.2 & 0.1 & 0.2 & 0.1 \\ -0.2 & -0.1 & -0.3 & -0.2 & -0.1 & -0.2 & -0.1 & -0.3 \end{pmatrix}$$

$$b = \begin{pmatrix} 0.5 \\ -0.5 \end{pmatrix}$$

§2.3核心组件:全连接层

全连接层

全连接层通过将卷积层或池化层生成的特征图展平为一维向量,并与权重矩阵进行全连接运算,整合全局特征,用于分类、回归等任务.

- **全连接操作**
  计算:$y = Wx + b$
- **计算过程**
  - $y_1 = 0.1 \cdot 6 + 0.2 \cdot 8 + \cdots + 0.1 \cdot 9 + 0.5 = 4.8$
  - $y_2 = (-0.2) \cdot 6 + (-0.1) \cdot 8 + \cdots + (-0.3) \cdot 9 + (-0.5) = -3.4$
  - 输出向量:$y = (4.8, -3.4)^T$
  - 应用Softmax:$\text{Softmax}(y) = (0.99, 0.01)^T$表示其为类型1的概率为99%
- **意义**
  - 全连接层整合特征图信息,输出分类概率
  - 权重矩阵W 学习特征与类别的映射关系

§2.4核心组件:正则化层

正则化层

正则化层通过在训练过程中引入约束或随机化机制,限制模型复杂度,防止过拟合,提升模型在陌生数据上的泛化能力.

- **正则化方式**
  - Dropout:在训练时随机弃置神经元,阻止模型过度依赖部分神经元
  - L2正则化:在损失函数中添加权重惩罚项,限制权重过大
  - Batch Normalization:标准化每一层的输入,稳定训练过程,间接减少过拟合
- **正则化效果**
  - 减少模型对训练数据噪声的敏感性,避免“记住”训练数据细节
  - 提高模型在测试数据上的性能,增强泛化能力
  - 以Dropout为例:在训练时,神经元以概率p被弃置,输出值为0
- **正则化优势**
  - 有效防止过拟合,尤其在深层网络或数据量不足时
  - 提高模型鲁棒性,适应多样化的输入数据

§2.4核心组件:正则化层

正则化层

正则化层通过在训练过程中引入约束或随机化机制,限制模型复杂度,防止过拟合,提升模型在陌生数据上的泛化能力.

- **数据设定**
  假设全连接层输入为展平后的特征向量为$x = (6, 8, 14, 16)^T$ 权重矩阵$W \in \mathbb{R}^{2\times 4}$,目标为二分类任务:

$$W = \begin{pmatrix} 0.1 & 0.2 & 0.1 & 0.3 \\ -0.2 & -0.1 & -0.3 & -0.2 \end{pmatrix} \quad b = \begin{pmatrix} 0.5 \\ -0.5 \end{pmatrix}$$

- **Dropout操作**
  设定Dropout概率$p = 0.5$,即每个神经元有50%的概率被弃置
  - 引入时间种子随机选择弃置的神经元,此处弃置第2和第4个元素.
  - 输入向量变为:$x' = (6, 0, 14, 0)^T$.
  - 在推理阶段,权重乘以 $1 - p = 0.5$以补偿训练时的.

§2.4核心组件:正则化层

正则化层

正则化层通过在训练过程中引入约束或随机化机制,限制模型复杂度,防止过拟合,提升模型在陌生数据上的泛化能力.

- **计算过程**
  - 训练时:$y' = Wx' + b$,计算:

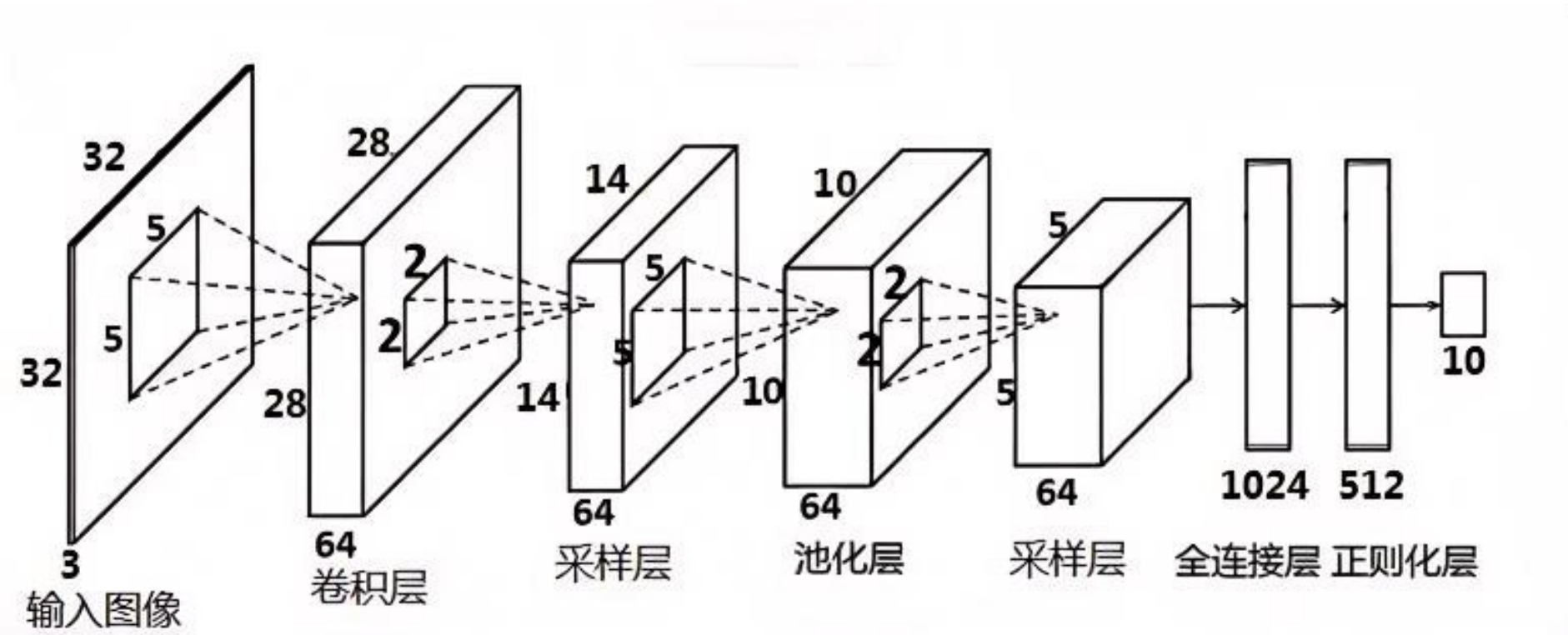
$$y'_1 = 0.1 \cdot 6 + 0.2 \cdot 0 + 0.1 \cdot 14 + 0.3 \cdot 0 + 0.5 = 2.3$$

$$y'_2 = (-0.2) \cdot 6 + (-0.1) \cdot 0 + (-0.3) \cdot 14 + (-0.2) \cdot 0 + (-0.5) = -5.9$$

    输出:$y' = (2.3, -5.9)^T$.
  - 推理时:使用缩放权重$Wt(1-p)$,保证输出一致性.
- **意义**
  - Dropout随机弃置神经元,迫使模型学习更鲁棒的特征组合.
  - 防止模型过拟合训练数据,提升在测试数据上的表现.

CNN的算法流程

CNN通过卷积层提取局部特征,结合池化层降低维度,全连接层整合信息,用于分类、检测等任务,同时可引入正则化层防止过拟合.



§4人工智能应用

AI图像识别

基于TensorFlow,CNN算法可用于二分类任务(区分AI生成图像与真实照片).以下将结合代码实际,分析其具体实现流程及特性.
特别注意的是,应用该代码需要确保Python环境在3.10及以前,否则无法键入稳定的TensorFlow框架,进而进行CNN.如果Python环境较新,可以选择创建虚拟环境.venv,这样可以确保在稳定运行TensorFlow的情况下不影响其它函数的使用.

- **数据预处理与增强**
  - **数据集加载**:使用`image_dataset_from_directory`从指定目录加载图像,尺寸统一为150×150,分为训练集(80%)和验证集(20%),批次大小为32
  - **数据增强**:通过`Random`系列函数实现水平翻转、旋转($\pm 20\%$)、放缩($\pm 20\%$)和对比度调整,增强数据多样性,防止过拟合
  - **归一化**:使用`Rescaling`层将像素值从[0,255]放缩到[0,1],提高训练稳定性

§4人工智能应用

- **模型架构**
  - **卷积层**:包含3个卷积层,分别使用32、64、128个3阶卷积核,激活函数为ReLU,提取多尺度特征
  - **池化层**:每个卷积层后接2阶池化窗口,步幅为2,减少特征图尺寸(约减半),降低计算量
  - **全连接层**:全连接层将特征图展平为一维向量,后接512单元的ReLU激活和1单元的Sigmoid激活,用于二分类
  - **Dropout正则化层**:在全连接层后添加`Dropout(0.5)`,随机丢弃50%神经元,防止过拟合
  - **优化器**:使用`Adam`优化器,学习率为0.001,适合快速收敛
- **模型编译与训练**
  - **训练设置**:训练10个周期(`epochs=10`),在训练集上拟合模型,验证集评估性能
  - **保存模型**:训练后保存为`ai_vs_photo_model.h5`，便于后续加载。
  - **推理函数**:`predict_image`加载单张图像，调整尺寸为150×150,归一化后预测,以0.5为阈值输出AI或Photo

CNN的算法流程

CNN通过卷积层提取局部特征,结合池化层降低维度,全连接层整合信息,用于分类、检测等任务,同时可引入正则化层防止过拟合.

- Python的技术部分如下所示.

```
model=Sequential()
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(150,150,3)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid'))
```

参考文献

[1] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks [J]. Communications of the ACM, 2017, 60(6): 84-90. DOI: 10.1145/3065386.
[2] Wang S Y, Wang O, Zhang R, et al. CNN-generated images are surprisingly easy to spot... for now [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S.l.]: IEEE, 2020: 8695-8704.

浅析CNN算法及其人工智能应用

结合 Alex,Wang 的研究 与 AI图像检测

侯东杨

北京林业大学 理学院

2025 年 6 月 5 日