International Journal of Knowledge and Language Processing
Volume 13, Number 2, Dec **2022**

AKLP International ©2010 ISSN 2191-2734
pp. 13–**27**

A Review of Residual Convolutional Networks

Wuwen Qiu
Peking University
No.24, Jinyuan Road, Daxing District, Beijing, China
wuwenqiu@stu.pu.edu.cn

ABSTRACT. *With the rapid development of deep learning methods in artificial intelligence, the learning ability of neural network models has also progressed with the increase of model size and has been able to challenge many complex human tasks. ResNet, a residual network, is a milestone in the development of deep neural networks, and its unique residual connectivity method makes deep model training possible. This study analyzes residual networks' construction, principles, and code implementation process. The research on domestic and foreign artificial intelligence applications applying the residual network and its variants in recent years is summarized. The application of residual networks in enterprise projects is also analyzed, and the opportunities of residual networks in today's deep learning model landscape are sorted out.*
Keywords: residual networks, code analysis, deep learning models

1. **Introduction.** Artificial Intelligence (AI) is an important area of computer science and is considered one of the three frontier technologies of the 21st century. Since 2006, with the emergence of "deep learning neural networks" proposed by Hinton[1], deep learning methods have achieved great success in the field of artificial intelligence, with remarkable results in various tasks such as image classification, machine translation, decision management, etc. It has become one of the most mainstream techniques in the field of artificial intelligence today. As many researchers in the field continue to innovate on deep learning neural networks, AI is widely used and productive in many disciplines, and researchers expect to use deep learning neural network methods to solve more complex and realistic human problems.

For the model to achieve better results in learning complex problems, the neural network must have a more powerful nonlinear representation, be able to fit more complex feature inputs and learn more complex variations. This means that the neural network must be supported by sufficiently high model complexity. The development in complexity of a neural network can depend on the increase in both width and depth dimensions, and the increase in depth is often less costly than the increase in width. Therefore, sufficient network depth becomes necessary for the model to be competent in complex tasks. How to

effectively increase the network depth becomes an important topic in the field of deep learning.

2. **Residual network background.** Before the introduction of the residual network ResNet [2] in 2015, the number of layers of most neural network models was mostly kept below 30. In the ImageNet competition in 2012, AlexNet [3] used dropout in the fully connected layer for the first time to reduce overfitting and used the Relu activation function to achieve an accuracy much higher than the second place. At this time, AlexNet has 8 layers, of which the first 5 are convolutional layers and the last 3 are fully connected. 2014 Oxford University team proposed the VGG model [4] developed from AlexNet, featuring multiple 3*3 convolutional kernels and 2*2 pooling layers, with two versions of 16 and 19 layers. In the same year, GoogLeNet [5] proposed the introduction of the Inception module to increase the network width to enhance the model complexity to avoid the negative effect of increasing the number of layers of the network, and GoogLeNet without fully connected layers has a depth of 22. The performance of the model in this period was severely limited by the depth of the model, and the optimization problem brought by deepening the number of layers of the network became the bottleneck of deep network training at that time. Increasing the depth of the model will bring about the phenomenon of Gradient Explosion or Gradient Extinction, which makes the performance of the model decrease instead of increase when the network is deepened and is therefore called the degradation phenomenon.

The ResNet residual network was first proposed by Kai-Ming He in his paper "Deep Residual Learning for Image Recognition" in 2015. The core idea of the residual network is that the output of the previous layer of the neural network is directly added to the input of the current layer to realize the learning of residuals by the network to achieve a better training effect, which can be expressed by the formula $x_{l+1} = x_l + F(x_l, W_l)$. The introduction of the residual learning module makes the residual network easier to optimize compared with the traditional CNN model so that it can obtain better learning performance by increasing the depth. The emergence of ResNet has solved the problem of the limited depth of the neural network model, making the maximum model depth reach an unprecedented 152 layers. With 152 layers of depth, ResNet achieved first place in the ImageNet detection, localization, and other tasks competition in 2015. The residual network was a breakthrough and a milestone in deep learning. It profoundly changed the architectural design of deep neural network models since then, and has gained wide application and improvement. The use of residual ideas can still be seen in the most mainstream Transformer[6], Bert[7] , and other models in the deep learning field today.

3. **Principle of residual network.**

3.1 **Residual connection.** The residual block is the core component of a residual network, which consists of two parts: direct mapping and residual connection. The direct mapping part generally contains two or more convolutional processes, similar to an ordinary convolutional block, and residual connectivity is the essential feature of residual blocks. Residual connectivity, also known as jump connectivity or short-circuit connectivity, is a process in which the input of the current block is directly added to the output of the final

layer through a cross-layer connectivity channel, and this processing can fuse features from different sensory fields. Residual connectivity is very effective in alleviating the problem of gradient explosion or gradient disappearance associated with deep model training and improves the performance and efficiency of the model while the network becomes deeper and wider. It can be said that residual connectivity makes the training of effective deep models possible.

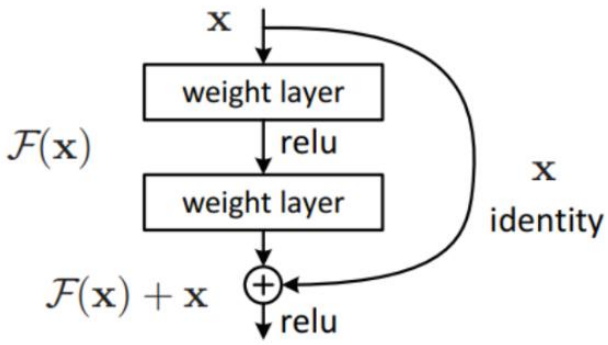

FIGURE 1. Schematic diagram of the residual connection

3.2 **Residual network structure.** There are various versions of residual networks with different depths, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet152, etc. Different versions of residual networks share some structural design, for example, the original input is first downsampled by a large 7*7 convolution kernel, all networks use four residual layers as the main body of the model and downsample the final output feature map by averaging pooling operation. But they differ in the design of the residual layer construction and the internal structure of the residual block.

The depths of ResNet-18 and ResNet-34 are relatively low, and their basic residual blocks consist mainly of two 3*3 convolutional layers. For models with more than 50 layers in-depth, small 1*1 convolutional layers are used at the head and tail of the block to control the depth of the feature map to cope with the high computational and parametric count problems caused by the excessive depth of the model. The number of convolutions contained in each residual layer varies for different depth models, as shown in the following comparison.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

FIGURE 2. Comparison of different depth residual networks

3.3 **Residual network improvement.** After the publication of ResNet, the field of deep learning has produced many more network architectures derived from ResNet. For example, DenseNet[8], proposed by Huang, G. et al in 2017, further extended the idea of residuals to interconnect the feature maps of all layers to form a more dense connection mechanism. ResNeXt[9] is a new version of ResNet proposed by Microsoft ASTRI. It incorporates the Inception multi-branching strategy on top of ResNet to increase the network width and further improve the complexity and performance of the network. WideResNet[10] also increases the network width on top of ResNet, but compared to ResNeXt which relies on the increase of the number of paths through the network, it achieves the width by expanding the number of filters in the convolutional layers. In the 2018 proposal SENet[11], the "Squeeze-and-Extraction" module is added based on ResNet, which can adaptively adjust the channel map weights to improve the network performance. In Yolo3 [12] DarkNet53, the residual connection is used extensively, and a convolutional layer with a step size of 2 is used instead of a pooling layer for downsampling to further mitigate the degradation problem. The accuracy of DarkNet53 with 53 layers in ImageNet is similar to that of ResNet-152.

4. **Residual network implementation code analysis.** This section explains the implementation code of the ResNet algorithm, the source code is from the official Pytorch repository "vision" on the GitHub platform, available at:
https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py.

4.1 **Two types of basic residual block source code analysis.**

4.1.1 **Analysis of the meanings of basic variables and parameters.**

(1) expansion: determine whether the number of convolutional kernels of the main branch has changed in the residual structure

(2) inplanes: input channels of residual blocks

(3) planes: the output channels of the residual block

(4) stride: the sampling interval of the convolution kernel after the input feature map

(5) downSample: whether to adjust the number of channels for down-sampling

(6) groups: the number of channel separations of group convolution, group=1 for normal convolution
(7) base_width : Number of base channels
(8) dilation: the spacing of the points of the convolution kernel in the dilated convolution, ordinary convolution when dilation=1
(9) norm_layer: whether there is a normalized layer

4.1.2 **BasicBlock class source code analysis.**

```
class BasicBlock(nn.Module):
    expansion: int = 1

    def __init__(self, inplanes, planes, stride=1, downsample=None,
                 groups=1, base_width=64, dilation=1, norm_layer=None):
        super().__init__()
        if norm_layer is None:
            norm_layer = nn.BatchNorm2d
        if groups != 1 or base_width != 64:
            raise ValueError("BasicBlock only supports groups=1 and base_width=64")
        if dilation > 1:
            raise NotImplementedError("Dilation > 1 not supported in BasicBlock")
        self.conv1 = conv3x3(inplanes, planes, stride)
        self.bn1 = norm_layer(planes)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = conv3x3(planes, planes)
        self.bn2 = norm_layer(planes)
        self.downsample = downsample
        self.stride = stride

    def forward(self, x: Tensor) -> Tensor:
        identity = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)

        if self.downsample is not None:
            identity = self.downsample(x)

        out += identity
        out = self.relu(out)

        return out
```

FIGURE 3. Comparison of different depth residual networks

BasicBlock is the basic residual block used to implement ResNet-18 and ResNet-34, two relatively shallow networks. The initialization function of this class defines the required components and adjustable properties in each basic residual block, including 3*3 convolution, normalization, ReLU activation function, downsampled property to decide whether to downsample, and stride property to adjust the sampling interval of the feature map. The “if” condition in the initialization function determines that BasicBlock does not support expanding convolution, group convolution, or increasing the network width.

The forward function in the BasicBlcok class shows how BasicBlock is constructed. Each basic residual block will be internally forward propagated in the order of 3*3 convolution, normalization, ReLU activation, 3*3 convolution, and normalization. The final block output is obtained by adding the output out variable of the current layer to the input identity variable of the representation block and then activating it again. Here is how the residual connection is implemented and the core feature of the residual block.

4.1.3 **Bottleneck class source code analysis.**

```
class Bottleneck(nn.Module):
    expansion: int = 4
    def __init__(self, inplanes, planes, stride=1, downsample=None,
                 groups=1, base_width=64, dilation=1, norm_layer=None):
        super().__init__()
        if norm_layer is None:
            norm_layer = nn.BatchNorm2d
        width = int(planes * (base_width / 64.0)) * groups
        self.conv1 = conv1x1(inplanes, width)
        self.bn1 = norm_layer(width)
        self.conv2 = conv3x3(width, width, stride, groups, dilation)
        self.bn2 = norm_layer(width)
        self.conv3 = conv1x1(width, planes * self.expansion)
        self.bn3 = norm_layer(planes * self.expansion)
        self.relu = nn.ReLU(inplace=True)
        self.downsample = downsample
        self.stride = stride

    def forward(self, x: Tensor) -> Tensor:
        identity = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)
        out = self.relu(out)

        out = self.conv3(out)
        out = self.bn3(out)

        if self.downsample is not None:
            identity = self.downsample(x)

        out += identity
        out = self.relu(out)

        return out
```

FIGURE 4. Comparison of different depth residual networks

The Bottleneck is the basic residual block used to implement the three deep networks ResNet-50, ResNet-101, and ResNet-152. Compared with BasicBottle, it has more tuning space and better scalability. For example, BottleNeck can achieve channel separation by adjusting the group parameter, increasing the channel width by setting the base_width parameter, and expanding the convolution by setting the dilation parameter. These processing methods exist mainly to reduce the number of parameters that need to be trained for deep network models and improve the model training efficiency.

The Bottleneck is the same as BasicBlock in terms of basic components except that a small 1*1 convolution kernel is added to compress the dimensionality, but there is a difference in the way the layers are connected. According to the structure shown by the forward function, the Bottleneck internally propagates forward in the order of 1*1 convolution, normalization, ReLU activation, 3*3 convolution, normalization, ReLU activation, 1*1 convolution, normalization, residual connection (block input and output summed), and ReLU activation again.

4.2 **Residual network model source code analysis.**

```
class ResNet(nn.Module):
    def __init__(self, block, layers, num_classes=1000, zero_init_residual=False,groups=1,
                 width_per_group=64, replace_stride_with_dilation=None, norm_layer=None):
        super().__init__()
        _log_api_usage_once(self)
        if norm_layer is None:
            norm_layer = nn.BatchNorm2d
        self._norm_layer = norm_layer

        self.inplanes = 64
        self.dilation = 1
        if replace_stride_with_dilation is None:
            replace_stride_with_dilation = [False, False, False]
        if len(replace_stride_with_dilation) != 3:
            raise ValueError(
                "replace_stride_with_dilation should be None "
                f"or a 3-element tuple, got {replace_stride_with_dilation}"
            )
        self.groups = groups
        self.base_width = width_per_group
        self.conv1 = nn.Conv2d(3, self.inplanes, kernel_size=7, stride=2, padding=3, bias=False)
        self.bn1 = norm_layer(self.inplanes)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)
        self.layer1 = self._make_layer(block, 64, layers[0])
        self.layer2 = self._make_layer(block, 128, layers[1], stride=2, dilate=replace_stride_with_dilation[0])
        self.layer3 = self._make_layer(block, 256, layers[2], stride=2, dilate=replace_stride_with_dilation[1])
        self.layer4 = self._make_layer(block, 512, layers[3], stride=2, dilate=replace_stride_with_dilation[2])
        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(512 * block.expansion, num_classes)

        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.kaiming_normal_(m.weight, mode="fan_out", nonlinearity="relu")
            elif isinstance(m, (nn.BatchNorm2d, nn.GroupNorm)):
                nn.init.constant_(m.weight, 1)
                nn.init.constant_(m.bias, 0)
        if zero_init_residual:
            for m in self.modules():
                if isinstance(m, Bottleneck) and m.bn3.weight is not None:
                    nn.init.constant_(m.bn3.weight, 0)
                elif isinstance(m, BasicBlock) and m.bn2.weight is not None:
                    nn.init.constant_(m.bn2.weight, 0)
```

FIGURE 5. ResNet model initialization source code

The initialization function _init_() in the ResNet class source code explains the building blocks required for ResNet and defines the initial parameters for some of the layers. In the input of _init_(), the layers parameter is an array that records the number of stacked layers for each residual layer from layer1 to layer4, the replace_stride_with_dilation parameter is an array that records the size of the dilated convolutional spacing of the three residual layers layer2, 3, and 4, and the number of input channels is initialized to 64. The most important part of the overall architecture of ResNet is the residual module of layer1 to layer4. Only layer1 is not processed for sampling interval and dilation convolution, while the other three layers will take down sampling and dilation convolution to reduce the amount of parameter computation, and the number of channels of the feature map will be increased to four times the initial one in the process.

```
def _forward_impl(self, x: Tensor) -> Tensor:
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)

    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)

    x = self.avgpool(x)
    x = torch.flatten(x, 1)
    x = self.fc(x)

    return x
```

FIGURE 6. ResNet _forward_impl function code

The _forward_impl function shows the order in which the layers are connected in the ResNet architecture. The initial image input is first processed by 7*7 convolution, normalization, ReLU activation, and maximum pooling, and the feature map size is only one-quarter of the initial image input. Then we enter the residual module, in which the four residual layers layer1 to layer4 are connected sequentially. The final fully connected layer transforms the number of feature channels in the final output to the number in the num_classes parameter, which is 1000 by default.

4.3 **ResNet input and output formats.**

4.3.1 **Input format.** The image format used by ResNet is a color image with a resolution of 224*224, and the input to the model is a tensor representation of such an image. Images can be converted into a tensor input of the shape (batch_size, 3, 224, 224) using torch.Tensor package. The figure below shows an example of a single image, i.e., an input tensor with a batch size of 1.

```
tensor([[[-0.0116, -0.0287,  0.0056,  ...,  0.1597,  0.1426,  0.1254],
         [-0.0629, -0.0458, -0.0287,  ...,  0.0912,  0.1254,  0.1597],
         [-0.0458, -0.0972, -0.0287,  ...,  0.0912,  0.1597,  0.1939],
         ...,
         [ 0.8447,  0.8618,  0.8447,  ...,  1.9920,  1.9920,  1.9749],
         [ 0.8618,  0.8618,  0.8789,  ...,  1.9578,  1.9578,  1.9578],
         [ 0.8961,  0.9132,  0.9303,  ...,  1.9578,  1.9749,  1.9578]],

        [[ 0.4853,  0.4678,  0.5028,  ...,  0.7129,  0.6954,  0.6779],
         [ 0.4328,  0.4503,  0.4678,  ...,  0.6429,  0.6779,  0.7129],
         [ 0.4503,  0.3978,  0.4678,  ...,  0.6429,  0.7129,  0.7479],
         ...,
         [ 0.4153,  0.4328,  0.4153,  ...,  1.9559,  1.9734,  1.9909],
         [ 0.4328,  0.4503,  0.4678,  ...,  1.9384,  1.9559,  1.9734],
         [ 0.5203,  0.5378,  0.5378,  ...,  1.9384,  1.9734,  1.9734]],

        [[ 0.5834,  0.5659,  0.6008,  ...,  0.9319,  0.9145,  0.8971],
         [ 0.5311,  0.5485,  0.5659,  ...,  0.8622,  0.8971,  0.9319],
         [ 0.5485,  0.4962,  0.5659,  ...,  0.8622,  0.9319,  0.9668],
         ...,
         [-0.2532, -0.2532, -0.2881,  ...,  1.7860,  1.8034,  1.8208],
         [-0.2707, -0.2707, -0.2532,  ...,  1.7337,  1.7511,  1.7685],
         [-0.1835, -0.1835, -0.1835,  ...,  1.7163,  1.7511,  1.7685]]])
```

FIGURE 7. Example of ResNet input tensor

Images with a resolution other than 224*224 can also be transformed to match the input tensor format by calling Pytorch's “transforms” preprocessing method. the Pytorch image preprocessing code is as follows.

4.3.2 **Output format.** The output format of ResNet is related to num_classes in the initialization parameters. Its output is a tensor of the shape (batch_size, num_classes). Using the default value of 1000 for this parameter as an example, the sample output graph of ResNet with a single image batch_size=1 is shown below.

```
tensor([[0.7175, 0.1302, 0.8103, ..., 0.0156, 0.8070, 0.9244]])
```

FIGURE 8. Example of ResNet input tensor

A row in the output tensor represents the prediction result of an image. Each row contains num_classes elements, indicating the probability that the image belongs to each class. For example, if the value of the first element in the first row is 0.7175, it means that the probability that the first image belongs to class 1 is 71.75%.

5. **Review of residual network applications.** The application of the ResNet algorithm is commonly found in intelligent systems in the fields of medical care, transportation, product quality monitoring, etc., specifically including systems for tasks such as image recognition and classification, audio recognition and classification, and future event prediction. At present, domestic research on the application of the ResNet algorithm mainly focuses on image recognition and classification systems in niche areas, such as adulterated mutton detection systems, gun species identification systems, etc. These segments often lack data sets suitable for training and testing deep learning models, and sometimes face the

problems of few data resources in the field and difficulty in sample collection. Therefore, researchers need to first build their sample libraries and expand them with data augmentation methods, and then adapt the traditional ResNet models by, but not limited to, modifying individual residual units, introducing other mechanisms such as SVM or attention mechanisms, changing the model architecture to achieve lightweight, etc. In addition, experiments on ResNet models with a different number of layers are required to determine the optimal number of ResNet layers on this task.

Foreign research on the application of the ResNet algorithm is more diversified and the degree of modification of the ResNet algorithm is greater. In addition to traditional image recognition and classification, ResNet has also been applied to tasks such as urban traffic flow prediction and human behavior prediction with an added temporal dimension, and 3D-ResNet, which can handle temporal and spatial features, is widely used in these systems that need to capture spatiotemporal features. The following are some specific applications of the ResNet algorithm in research.

Yikai Wen, Le Chen, and Yaqiong Fu (2022) designed a ResNet18 classification network-based beep tone recognition system[13]. The system first uses a sound card and a microphone for beeping tone signal radio, then preprocesses the tone signal by coefficient comparison method for noise reduction, and transforms the processed signal into a signal time-frequency feature map to ResNet18 network model for analysis and classification, and the system recognition accuracy reaches 97.5%. The system contains functional modules such as signal feature display, signal acquisition parameter setting, recognition object selection, data communication, and data persistence in functional design. Min-Ling Zhu, Liang-Liang Zhao, and Shou-Jie (2022) studied and implemented an intelligent cattle face recognition and detection system based on the CNN-ResNet-SVM model[14]. To solve the problems of less cow face data and high data noise, ResNet and SVM are introduced into the traditional CNN to achieve faster training speed, higher recognition rate, and stronger generalization than the traditional CNN. The intelligent cow face recognition system experimented with a specific model of cell phone camera shooting at the APP side. The experimental results proved that the fastest recognition speed is around 128ms and the recognition accuracy is above 95.1%. Dongyu He, and Rongguang Zhu (2022) developed a smartphone application based on an attention mechanism combined with an inverted residual network (CBAM-Invert-ResNet)[15] for fast and accurate detection of lamb adulteration. The training dataset is firstly constructed by collecting original cell phone images of different parts of lamb, pork, and adulterated lamb, and augmented with data enhancement. Then the original residual network is replaced with an inverted residual structure to improve the model convergence speed and reduce the parameters, and then the feature weights are reassigned by an attention mechanism. The trained CBAM-Invert-ResNet50 model has a 61.64% less number of parameters and 61.59% less size than ResNet50, and the classification accuracy of the mixed-site dataset is at 92.96%. The model was deployed on mobile for detection, and the detection time was about 0.3 s per mobile image. Jiaxin Ling and Yonghua Xie (2022) built their sample library of

defective wood panels and expanded the dataset with data enhancement methods[16], expanding the sample library to 10678 images. The researchers divided the sample library into a training set, validation set, and test set, and trained various network models such as VGG, GoogleNet, and ResNet with different layers and tested them on the test set for classifying wood panel defects. The researchers also improved the residual unit by considering ReLU as a pre-activation of the weight layer to generate a new residual unit. The experimental results demonstrate that the improved residual network achieves 98.63% classification accuracy at 50 layers and is most suitable for defective wood panel classification. Zhou Zhifei, Wu Jinlong (2022) et al. established a firearms image recognition system based on multi-task cascaded residual networks for the problem of low efficiency of firearms species recognition[17]. The researchers first constructed the gun dataset based on the gun image acquisition specification and image enhancement specification construction. To achieve stable network training with few samples, a four-stage hierarchical gun image recognition and retrieval model is used to achieve hierarchical recognition and retrieval from coarse to fine. The network model uses ResNet18 as the basic building block to obtain advanced retrieval features by combining the output of the four-stage Softmax loss function. The model achieves 61.12% and 95.28% recognition accuracy in Rank-1 and Rank-20, respectively.

Huan Zhang, Liangxiao Jiang, et al. (2021) proposed a new model based on the standard ResNet, a "Cost-Sensitive Residual Convolutional Network" (CS ResNet), which is used to perform cosmetic defect detection on printed circuit boards (PCBs). [18]CS ResNet is optimized by minimizing the weighted cross-entropy loss function. CS ResNet achieves the highest accuracy (0.89-0.91) and the lowest classification error cost on the real PCB appearance defect dataset. The lowest classification error cost. Yoshiki Kakamu and Kazuhiro Hotta (2022) used 3D-ResNet[19] capable of processing temporal and spatial features to identify and predict human behavior in video images with high accuracy to help humans in the future of medical care. To improve the feature representation of traditional 3D ResNet for small actions, the authors propose a method to extract features by loops in residual blocks based on convolutional LSTM. The model of this loop mechanism exhibits higher accuracy than the traditional 3D-ResNet in UCF101, Kinetics-400 and HMDB-51 datasets.Y. Mao, Y. Zeng, et al. (2022) proposed a model consisting of ResNet and Conformer backbone network (SEKD-RCnet) and its two variants SED RCnet and SSL RCnet consisting of an integrated sound event localization and detection (SELD) system[20]. The system achieved significant improvements over the baseline in the L3DAS22 challenge. Robert Turnbull (2022) used the Resnet-based 3D convolutional neural network Cov3d for scanning chest CT to detect the presence and severity of noncoronary pneumonia. The model was trained on the COV19-CT-DB dataset and manually labeled by experts and obtained a Macro-f1 score of 87.87 on the test set for new coronary pneumonia detection. Rui He et al. (2022) developed a spatiotemporal 3D multi-scale ConvLSTM Resnet network and applied it to an intelligent transportation system (ITS)[21] to accurately predict the future traffic flow. The system uses 3D

DenseNet to capture the spatiotemporal information of traffic frames by considering the traffic data slices at each moment as "traffic frames", and introduces ConvLSTM Resnet to solve the problem that traditional Resnet cannot capture the spatial correlation over long distances. The model outperforms state-of-the-art methods for citywide traffic flow prediction and shows good generalizability for the task of predicting passenger travel demand.

6. **Application of residual network in enterprise practice.** In the current AI field, large-scale pre-trained models represented by BERT and GPT are just at the right time. By pre-training on a large amount of general data, the generality of the model is greatly improved, and on this basis, the model can be fine-tuned for downstream tasks to obtain the best performance. "Pre-training" plus "fine-tuning" has become the dominant paradigm in AI applications today. Even in this environment, ResNet models can still play an irreplaceable role in specific scenarios. In scenarios such as wearable devices and in-vehicle devices, where there are limitations on computing power and energy consumption, it is more difficult to deploy large models. At this time, ResNet architecture models can take advantage of its simplicity and lightweight. In addition, in the field of security, health care, and other sample data difficult to collect, large-scale pre-trained model solutions are difficult to use, while the direct use of ResNet models is more suitable for learning under such low training data conditions, and can effectively save the cost of computing and manual data labeling.

With the outbreak of the new crown epidemic in 2020, Didi has developed a mask recognition and prevention system, aiming to protect the health of drivers and passengers. The face recognition module of this system contains a mask attribute recognition model, and this model is improved based on ResNet50. Using the feature that masks are relatively stable in the face position, an attention learning mechanism is introduced based on ResNet50 to enhance the feature extraction of the mask region and enable better detection of difficult samples. This technology of DDT was deployed in the pre-departure intelligent exit system and in-vehicle devices during the epidemic, and is open and open source to the whole society. The model source code is available in DiDi's open-source repository at github.com/didi/maskdetection.

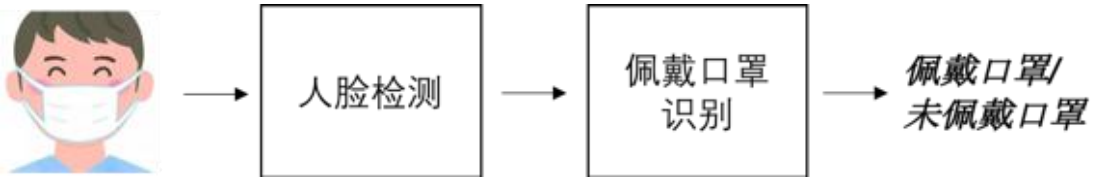

FIGURE 9. Overall block diagram of face-wearing mask recognition

On International Day of the Deaf in September 2020, Tencent Multimedia Fusion Lab opened the "Tian Lai AI" audio technology, which aims to create a cochlear implant with deep learning technology for people with hearing impairment and improve the hearing

experience of people with hearing impairment. The technology uses a residual network structure to process the collected noise, enabling deep-learning model noise reduction on a low-power cell phone terminal. The average speech recognition rate of the cochlear implant deployed with the Celestial AI technology reached 96.28%.

In February 2021 ByteDance AVG (Advanced Video Team) proposed DAM, a filter built by the residual convolutional neural network, which aims to reduce distortion in the video compression process. AVG put the technology into a BVC encoder and applied it in video processing of apps such as Jitterbug and Today's Headlines to bring users a higher quality playback experience.Figure6.1

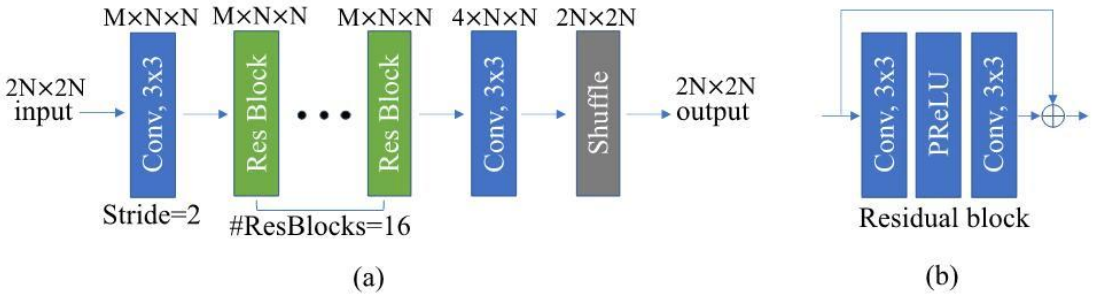

FIGURE 10. Deep convolutional neural network based on residual unit stacking in DAM

DiDi's AR real-world navigation product aims to solve the problem of intricate internal routes in large indoor locations such as airports, stadiums, and subway stations. The orientation estimation module of this AR navigation product uses a model called heading-confidence, which uses ResNet and LSTM as the framework. Among them, ResNet is responsible for regressing the walking speed, while LSTM is responsible for regressing the walking orientation. The product is currently online in 24 airports, shopping malls, and train stations in Zhengzhou, Shenzhen, and Tokyo, Japan, and according to the data, it can help users save nearly 25% of their time.

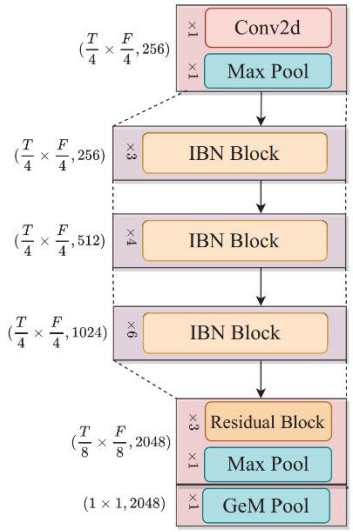

FIGURE 11. ResNet-IBN architecture diagram

2022 ByteDance Volcano Speech Team released its latest music retrieval system ByteCover2. The system mainly serves the cover recognition task (CSI) and employs a multi-task learning paradigm based on residual networks jointly with the ResNet-IBN model. the ResNet-IBN model is capable of extracting robust and discriminative vector representations from audio inputs. ByteCover2 achieves SoTA performance on the Da-Tacos test dataset that far exceeds that of other schemes.

REFERENCES

[1] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778)

[3] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[4] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[5] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[6] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[8] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp.

4700-4708).

[9] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).

[10] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.

[11] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

[12] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.

[13] Wen Yikai, Chen Le, Fu Yaqiong. Design of abnormal electronic beep recognition system based on wavelet denoising and ResNet classification network [J]. Journal of Metrology, 2022,43 (11): 1486-1491

[14] Zhu Minling, Zhao Liangliang, and He Shoujie. Research and Implementation of a Cow Face Recognition System Model Combining CNN with SVM and ResNet [J]. Journal of Chongqing University of Technology (Natural Science), 2022,36 (07): 155-161

[15] He Dongyu, Zhu Rongguang, Fan Binbin, Wang Shichang, Cui Xiaomin, Yao Xuedong. Construction of adulterated mutton classification detection system based on inverted residual network and attention mechanism [J]. Journal of agricultural engineering, 2022,38 (20): 266-275

[16] Ling Jiaxin, Xie Yonghua. Application of residual neural network model in defect classification of wooden boards [J]. Journal of Northeast Forestry University, 2021,49 (08): 111-116. DOI: 10.13759/J.CNKI.DLXB.2021.08.021

[17] Zhou Zhifei, Wu Jinlong, Li Yiyi, Jia Lipang, Shen Yujie, Zhang Gang, Cui Bin. A Gun Image Recognition System Based on Multi task Cascaded Residual Network [J]. Computer Engineering, 2022,48 (01): 214 219. DOI: 10.19678/J.ISSN1000 3428.0059820

[18] Huan Zhang, Liangxiao Jiang, Chaoqun Li, CS-ResNet: Cost-sensitive residual convolutional neural network for PCB cosmetic defect detection, Expert Systems with Applications, Volume185,2021,115673, ISSN09574174.

[19] Y. Kakamu and K. Hotta, "Predicting Human Behavior Using 3D Loop ResNet," 2022 26th International Conference on Pattern Recognition (ICPR), Montreal, QC, Canada, 2022, pp. 3259-3264, doi: 10.1109/ICPR56361.2022.9956136.

[20] Y. Mao, Y. Zeng, H. Liu, W. Zhu and Y. Zhou, "ICASSP 2022 L3DAS22 Challenge: Ensemble of Resnet-Conformers with Ambisonics Data Augmentation for Sound Event Localization and Detection," ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, Singapore, 2022, pp. 9191-9195, doi: 10.1109/ICASSP43922.2022.9746673.

[21] Rui He, Yanbing Liu, Yunpeng Xiao, Xingyu Lu, Song Zhang, Deep spatio-temporal 3D densenet with multiscale ConvLSTM-Resnet network for citywide traffic flow forecasting, Knowledge-Based Systems, Volume 250, 2022, 109054, ISSN 0950-7051.