# Squeeze-and-Excitation Networks

Jie Hu , Li Shen , Gang Sun

**摘要**

卷积神经网络建立在卷积运算的基础上，通过在局部接受域内融合空间和通道信息来提取信息特征。为了提高网络的表征能力，许多现有的工作已经表明了增强空间编码的好处。在这项工作中，我们专注于通道，并提出了一种新的架构单元，我们称之为“挤压-激励”(SE) 块，通过显式建模通道之间的相互依赖性，自适应地重新校准通道方面的特征响应。我们证明，通过将这些块堆叠在一起，我们可以构建在具有挑战性的数据集上非常好地泛化的 SENet 架构。至关重要的是，我们发现 SE 块以很少的计算成本为现有最先进的深度架构带来了显著的性能改进。SENets 构成了我们的 ILSVRC 2017 分类提交的基础，该分类提交获得了第一名，并将前 5 名的误差显著降低到 2.251%，相对于 2016 年的获奖作品实现了 25% 的相对改进。

**关键词：**Squeeze；Excitation；attention；

## 1 引言

随着卷积神经网络在图像分类领域的崭露头角，CNN 已被证实是解决各种视觉任务的有效模型，其对于每个卷积层，学习一组过滤器来表达输入通道的局部空间连接模式，对于 CNN 网络来说，其核心计算是卷积算子，其通过卷积核从输入特征图学习到新特征图，卷积是对一个局部区域进行特征融合，这包括空间上（W 和 H 维度）以及通道间（C 维度）的特征融合，而对于卷积操作，很大一部分工作是提高感受野，即空间上融合更多特征，或者是提取多尺度空间信息。目前已经有很多工作在空间维度上来提升 CNN 的性能，从这个角度出发，网络可以从特征通道之间层面来考虑去提升性能。SENet 网络更加关注 channel 之间的关系，希望模型可以自动学习到不同 channel 特征的重要程度。具体来说，就是通过学习的方式来自动获取到每个特征通道的重要程度，然后依照这个重要程度去提升有用的特征并抑制对当前任务用处不大的特征。SENet 通过显式地建模卷积特征通道之间的相互依赖来提高网络的表示能力，提出了一种机制，允许网络执行特征重新校准，通过这种机制，能够充分的可以学习使用全局信息来选择性地强调信息丰富的特征，使得 CNN 能够更加综合提取图像的特征能力。

## 2 相关工作

### 2.1 深层架构

广泛的工作已经表明，以一种简化深度特征学习的方式重构卷积神经网络的架构可以显著提高性能[1]。VGGNets 和 Inception[2]模型证明了随着深度的增加可以获得的性能，在 ILSVRC 2014 比赛上的结果可以得到深层网络显著优于之前的方法。批处理归一化通过插入单元来调节层输入，稳定学习过程，从而实现更深入的进一步实验。不仅如此引入 ResNet 中的残差块能够使得整个网络层数变得更深而不用担心网络变深引起的梯度消失问题[3]，从而训练更深层次的网络是有效的。

### 2.2 注意力门控机制

注意力已被证明可以提高一系列任务的性能[4]，从图像中的处理到基于序列的模型。它通常与门控函数 (例如 softmax 或 sigmoid) 和顺序技术相结合实现，在这些应用程序中，它通常被用在一个或多个层之上，表示更高级的抽象，用于模式之间的适应，论文中提出的 se 块是一种轻量级的注意力门控机制，专门用于以计算高效的方式建模通道关系，并旨在增强整个网络模块的表示能力。

## 3 本文方法

### 3.1 本文方法概述

此部分对本文将要复现的工作进行概述 SE 模块的灵活性在于它可以直接应用现有的网络结构中。这里以使用 SE-block 块嵌入 ResNet 为例。对于 ResNet，SE 模块嵌入到残差结构中的残差学习分支中，首先使用 global average pooling 作为 Squeeze 操作，将全局空间信息压缩到通道特征图中，得到的输出可以被解释为局部特征图的集合，这些局部通道特征图的统计信息表达了整个图像。而 Squeeze 操作旨在完全捕获通道依赖性，并自适应重新校准为了利用挤压操作中聚集的信息。最后将经过 Squeeze 操作和 Excitation 操作的网络得到的输出通过 Scale 操作将各个通道分别乘上聚集的信息，得到最终输出。具体如图 1所示：

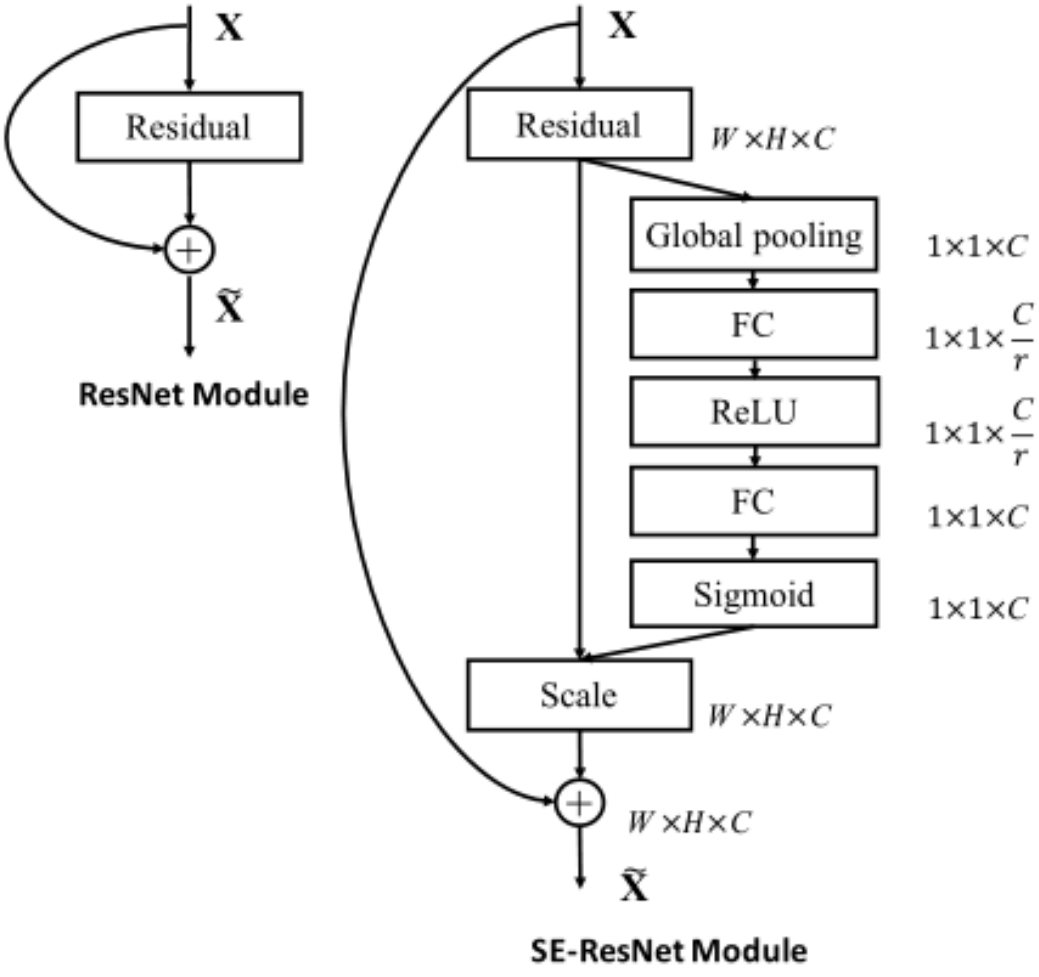


图 1: Se-resnet 结构图

### 3.2 Squeeze 模块

顺着根据空间维度来进行特征压缩,将每个二维的特征通道变成一个实数（即 C*H*W——C*1*1），这个实数某种程度上具有全局的感受野，并且输出的维度和输入的特征通道数相匹配。它表征着在特征通道上响应的全局分布，而且使得靠近输入的层也可以获得全局的感受野，其中 Squeeze 模块使用 global average pooling 作为 Squeeze 操作。形式化表示如图 2所示：

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} u_c(i, j).$$

图 2: Squeeze 操作

### 3.3 Excitation 模块

该操作由两个 Fully Connected 层组成一个 Bottleneck 结构去建模通道间的相关性，并输出和输入特征同样数目的权重。首先将特征维度降低到输入的 1/16，然后经过 ReLU 激活后再通过一个 Fully Connected 层升回到原来的维度。这样做比直接用一个 Fully Connected 层的好处在于：1）具有更多的非线性，可以更好地拟合通道间复杂的相关性；2）极大地减少了参数量和计算量。然后通过一个 Sigmoid 的门获得 0 1 之间归一化的权重.Sequeeze 操作得到了全局描述特征，因此需要 Excitation 操作来抓取 channel 之间的关系。这个操作需要满足两个准则：首先要灵活，它要可以学习到各个 channel 之间的非线性关系；第二点是它必须学习一个非互斥的关系，希望多个通道特征被加强，而不是像 one-hot 那种形式，只加强某一个通道特征。基于此，这里采用了简单的带有 sigmoid 激活函数的门控机制，形式化描述如图 3所示：

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z}))$$

图 3: Excitation 操作

### 3.4 Scale 模块

最后通过一个 Scale 的操作来将归一化后的权重加权到每个通道的特殊上。除此之外，SE 模块还可以嵌入到含有 skip-connections 的模块中。上右图是将 SE 嵌入到 ResNet 模块中的一个例子，操作过程基本和 SE-Inception 一样，只不过是在 Addition 前对分支上 Residual 的特征进行了特征重标定。如果对 Addition 后主支上的特征进行重标定，由于在主干上存在 scale 操作，在网络较深 BP 优化时就会在靠近输入层容易出现梯度消散的情况，导致模型难以优化, 形式化描述如图 4所示

$$\widetilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, s_c) = s_c \cdot \mathbf{u}_c$$

图 4: Scale 操作

# 4 复现细节

## 4.1 与已有开源代码对比

此部分为必填内容。如果没有参考任何相关源代码，请在此明确申明。如果复现过程中引用参考了任何其他人发布的代码，请列出所有引用代码并详细描述使用情况。同时应在此部分突出你自己的工作，包括创新增量、显著改进或者新功能等，应该有足够差异和优势来证明你的工作量与技术贡献。

在复现代码中，我的工作有，首先搭建 SEblock 块，并且在此模块中，首先搭建 Squeeze 操作部分的代码，Squeeze 使用全局平局池化对上一层的输出进行特征提取，实现过程如图 5所示：

```
# 论文核心 SE Block，这里称为 SE layer
class SELayer(nn.Module):
    def __init__(self, channel, reduction=16):
        super(SELayer, self).__init__()
        # 这里是squeeze操作，作全局平局池化，将通道之间的信息进行融合
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
            # Linear: from channel to channel/16
            nn.Linear(channel, channel // reduction, bias=False),
            # ReLu: 进行一次激活函数
            nn.ReLU(inplace=True),
            # Linear: from channel/16 to channel
            nn.Linear(channel // reduction, channel, bias=False),
            # Sigmoid: 激活到0-1，代表每个通道的重要性
            nn.Sigmoid()
        )
```

图 5: se-block 块示意图

接着是 Se-resnet 块，在这里我搭建了 Resnet-20 和 Resnet-50 层网络结构的残差网络，其中 BN 就是通过方法将该层特征值分布重新拉回标准正态分布，特征值将落在激活函数对于输入较为敏感的区间，输入的小变化可导致损失函数较大的变化，使得梯度变大，避免梯度消失，同时也可加快收敛。在 CIFAR-10 数据集中 Resnet 的大部分结构是 1 层 conv+n 个 block+1 层 fc, 因此在 resnet-20 中 n 应设置为 3，而一个 block 有 6 层卷积层，而对于 Resnet-50 层网络结构中则是 Bottleneck-block 块，其使用了 1×1 卷积层，1×1 卷积层的优势是在更深的网络中，用较小的参数量处理通道数很大的输入，Se-resnet 代码如图 6所示：

```python
# 这里是CifarSEResNet模块
class CifarSEResNet(nn.Module):
    # 其中block参决定是BasicBlock
    def __init__(self, block, n_size, num_classes=10, reduction=16):
        super(CifarSEResNet, self).__init__()
        self.inplane = 16
        self.conv1 = nn.Conv2d(
            3, self.inplane, kernel_size=3, stride=1, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(self.inplane)
        self.relu = nn.ReLU(inplace=True)
        # layer1中stride=1, layer2-4中stride=2
        self.layer1 = self._make_layer(
            block, 16, blocks=n_size, stride=1, reduction=reduction)
        self.layer2 = self._make_layer(
            block, 32, blocks=n_size, stride=2, reduction=reduction)
        self.layer3 = self._make_layer(
            block, 64, blocks=n_size, stride=2, reduction=reduction)
        self.avgpool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Linear(64, num_classes)
        self.initialize()
```

图 6: se-resnet 块示意图

搭建好网络模型后对该模型进行训练，训练轮数为 300 轮，学习率为 0.1，batch-size 为 64，使用动量发对梯度下降算法进行优化，训练部分代码如图 7所示：

```python
============================ step 5/5 训练 ============================
loss_rec = {"train": [], "valid": []}
acc_rec = {"train": [], "valid": []}
best_acc, best_epoch = 0, 0
stare_time = time.time()
for epoch in range(start_epoch + 1, MAX_EPOCH):

    # 训练(data_loader, model, loss_f, optimizer, epoch_id, device, max_epoch)
    loss_train, acc_train, mat_train = ModelTrainer.train(train_loader, se_resnet_model, criterion, optimizer, epoch,
    loss_valid, acc_valid, mat_valid = ModelTrainer.valid(valid_loader, se_resnet_model, criterion, device)
    print("Epoch[{:0>3}/{:0>3}] Train Acc: {:.2%} Valid Acc:{:.2%} Train loss:{:.4f} Valid loss:{:.4f} LR:{}".format(
        epoch + 1, MAX_EPOCH, acc_train, acc_valid, loss_train, loss_valid, optimizer.param_groups[0]["lr"]))

    scheduler.step()  # 更新学习率
    # 绘图
    loss_rec["train"].append(loss_train), loss_rec["valid"].append(loss_valid)
    acc_rec["train"].append(acc_train), acc_rec["valid"].append(acc_valid)
    if epoch == MAX_EPOCH-1: print('==============train acc================')
    show_confMat(mat_train, class_names, "train", log_dir, verbose=epoch == MAX_EPOCH-1)
    if epoch == MAX_EPOCH-1: print('==============valid acc================')
    show_confMat(mat_valid, class_names, "valid", log_dir, verbose=epoch == MAX_EPOCH-1)

    plt_x = np.arange(1, epoch+2)
    plot_line(plt_x, loss_rec["train"], plt_x, loss_rec["valid"], mode="loss", out_dir=log_dir)
    plot_line(plt_x, acc_rec["train"], plt_x, acc_rec["valid"], mode="acc", out_dir=log_dir)
```

图 7: 训练代码示意图

由于只加入传统的 SE-block 块虽然能够提升 CNNs 的网络性能，提高最后图片分类的准确率，但是 SE-block 其中的两个全连接层却仍引入了一定量的参数，这对于整个网络来说是不好的，同时增加了网络复杂性，因此可以将特征图分为不同组，然后对这些组分别进行卷积，同时也需要建模在不同跨通道之间的信息联系，因此引入一个全局权重参数来作为通道之间的注意力，做到共享参数这样的话可以在不增加 FC 参数量的情况下也同样带来性能的提升，这对于网络的鲁棒性是很重要的。

### 4.2 实验环境搭建

(1) 安装 anaconda,anaconda 是一个开源的 Python 发行版本，其包含了 conda、Python 等 180 多个科学包及其依赖项，最新的版本自带 Python3。(2) 安装完 anaconda 之后，打开 Anaconda Prompt，输入 conda create -n py38 创建一个虚拟环境，并且使用命令 conda activate py38 进入刚刚创建好的环境。(3) 安装 Pytorch、Cuda、matplotlib 等必要工具包，选择的 Pytorch 版本为 1.2.0，cuda 版本为 10.2，并且确保版本相互匹配，否则会因为版本不匹配原因导致程序无法运行。(4) 下载 Pycharm 专业版，并且使用 Pycharm 连接服务器，使程序在服务器的 GPU 上训练以加快训练速度。

### 4.3 创新点

为了消除 SE-block 中两个 FC 层增加的参数量，将特征图分为不同组，然后对这些组分别进行卷积，同时也需要建模在不同跨通道之间的信息联系，因此引入一个全局权重参数来作为通道之间的注意力，做到共享参数这样的话可以在不增加 FC 参数量的情况下也同样带来性能的提升，这对于网络的鲁棒性是很重要的。

## 5 实验结果分析

本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。对于 resnet20 网络模型在 CIFAR-10 上训练过程中的 Loss 下降图和准确率图如图 13所示：

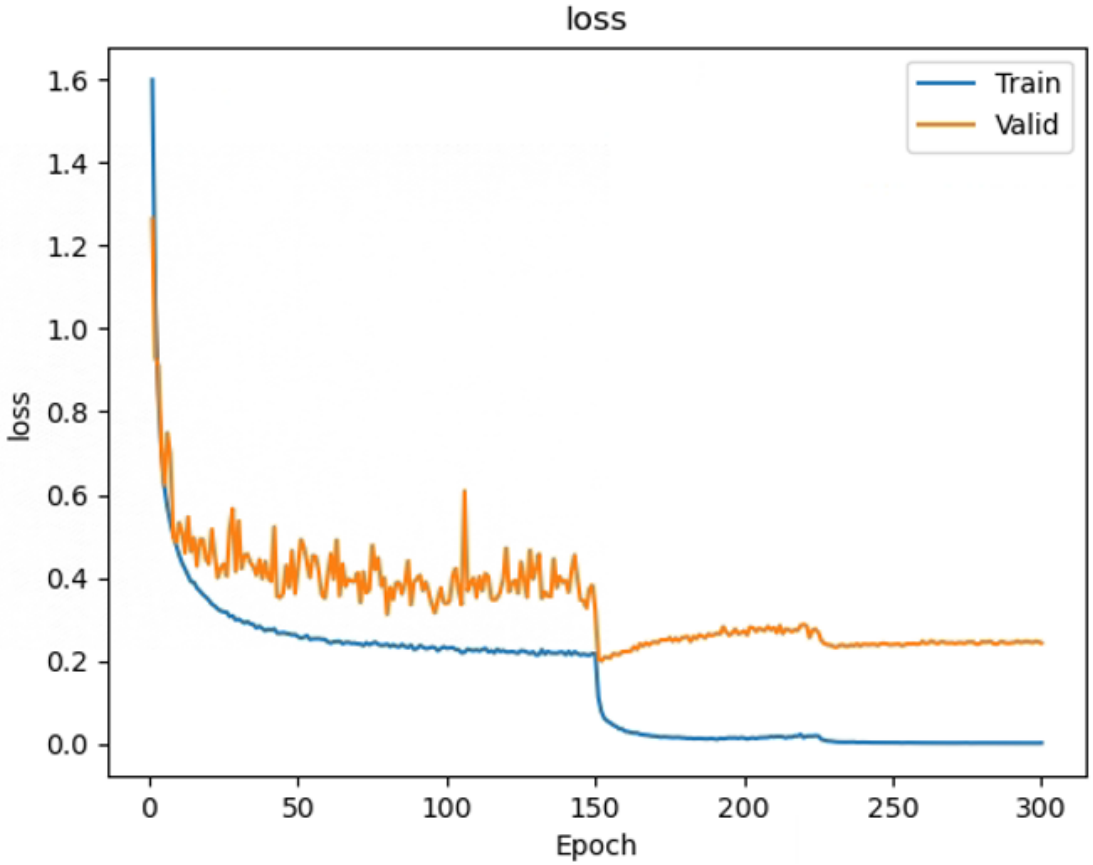


图 8: resnet20-loss

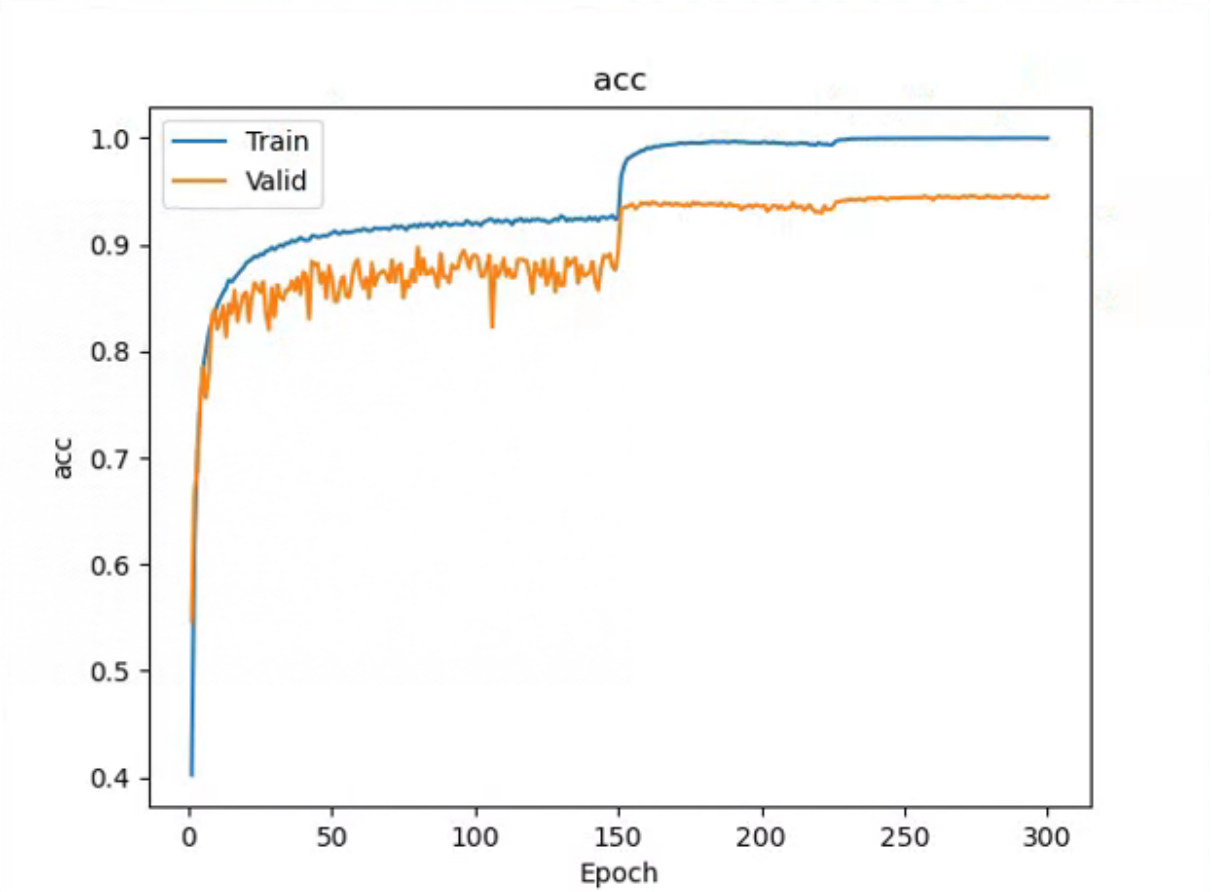


图 9: resnet20-acc

由上图可得，测试集中在前 150 轮时的 Loss 值略微震荡下降，并在该点急剧下降，说明跳出了局部最小值点在 150 轮之后在 0 处逐渐收敛，因此可以看出动量法优化梯度下降可以缓解震荡并且加速收敛，由于使用了 resnet 深度残差网络，即使网络层数变深也不会导致梯度消失，因此 Loss 能够不断收敛。

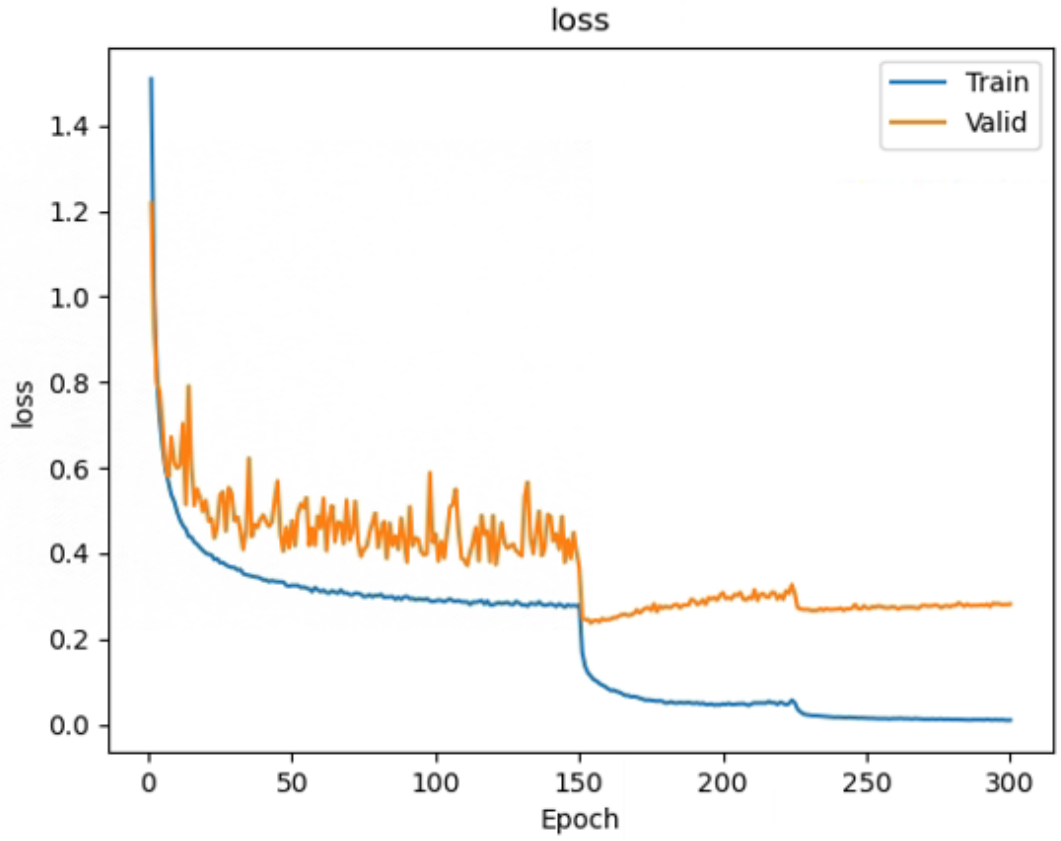


图 10: se-resnet20-loss

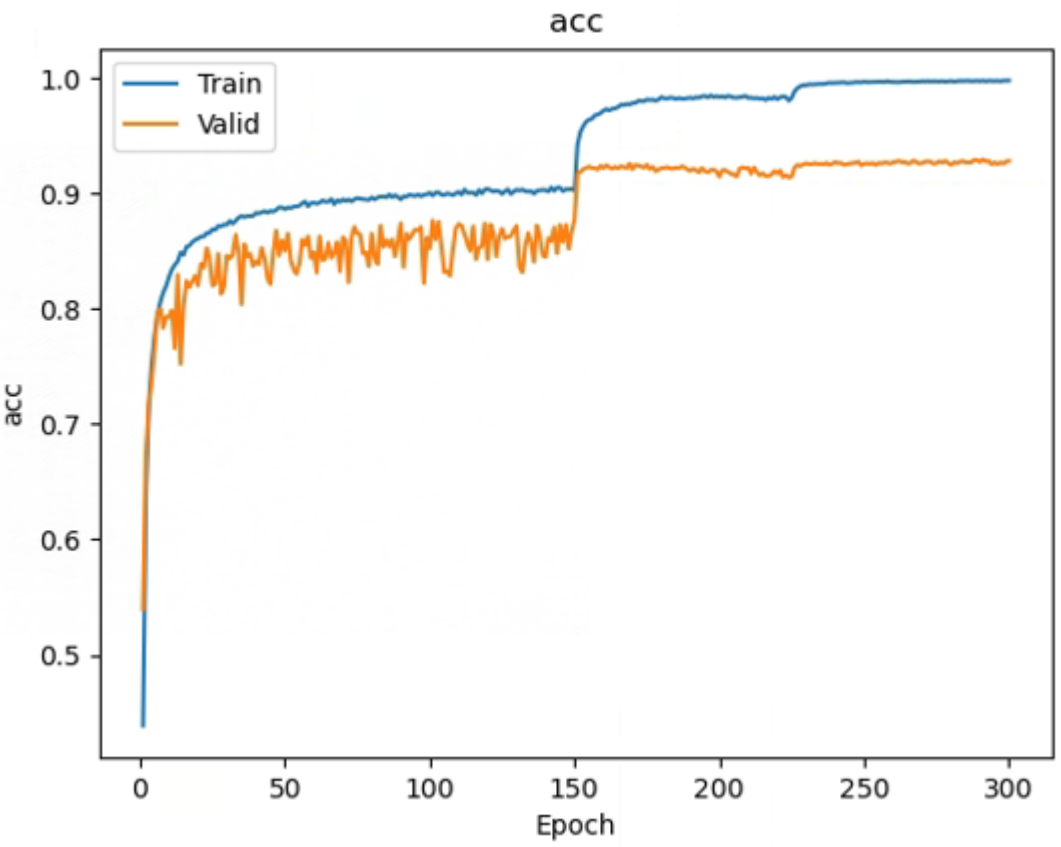


图 11: se-resnet20-acc

由上图可得，测试集中在前 150 轮的正确率逐步达到 90 左右，加入了 SE-block 块的 resnet 网络收敛速度较普通 resnet 网络收敛速度稍快，并且 se-resnet 在测试集上的测试正确率也比 resnet 的正确

率高，因此可以看出，SE-block 块的操作有助于网络性能的提升，并且能够在差别不大的运行时间内达到更好的效果。

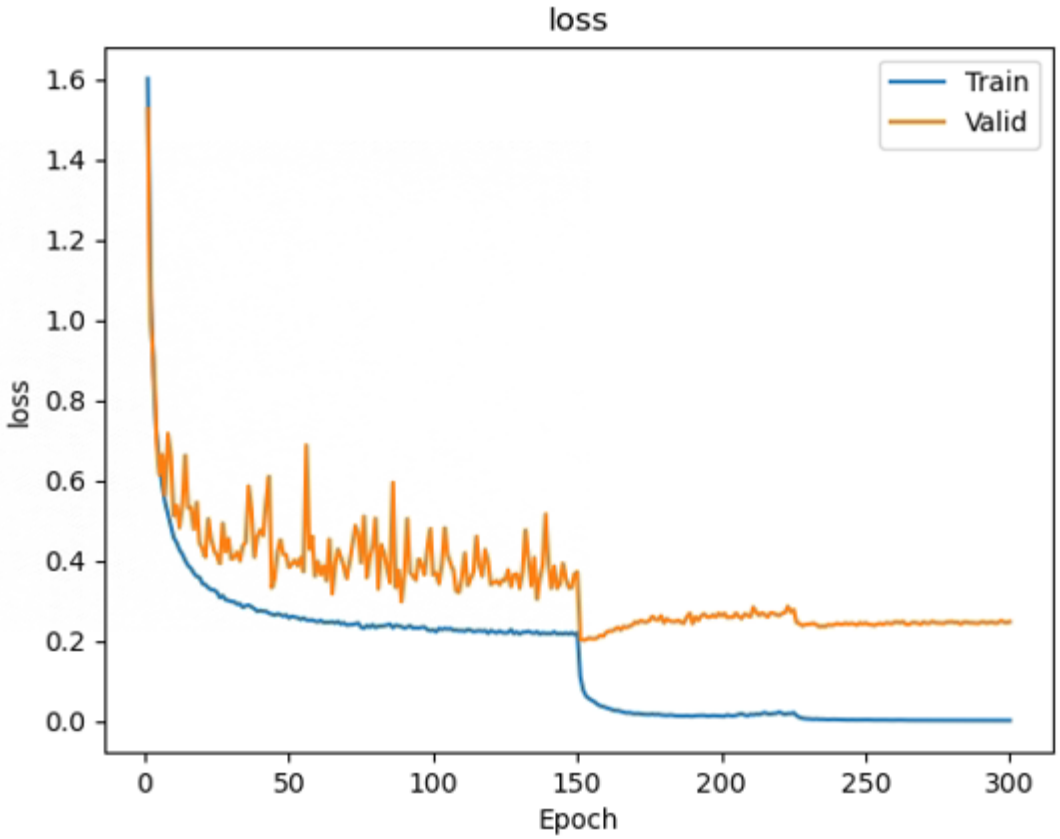


图 12: se-resnet56-loss

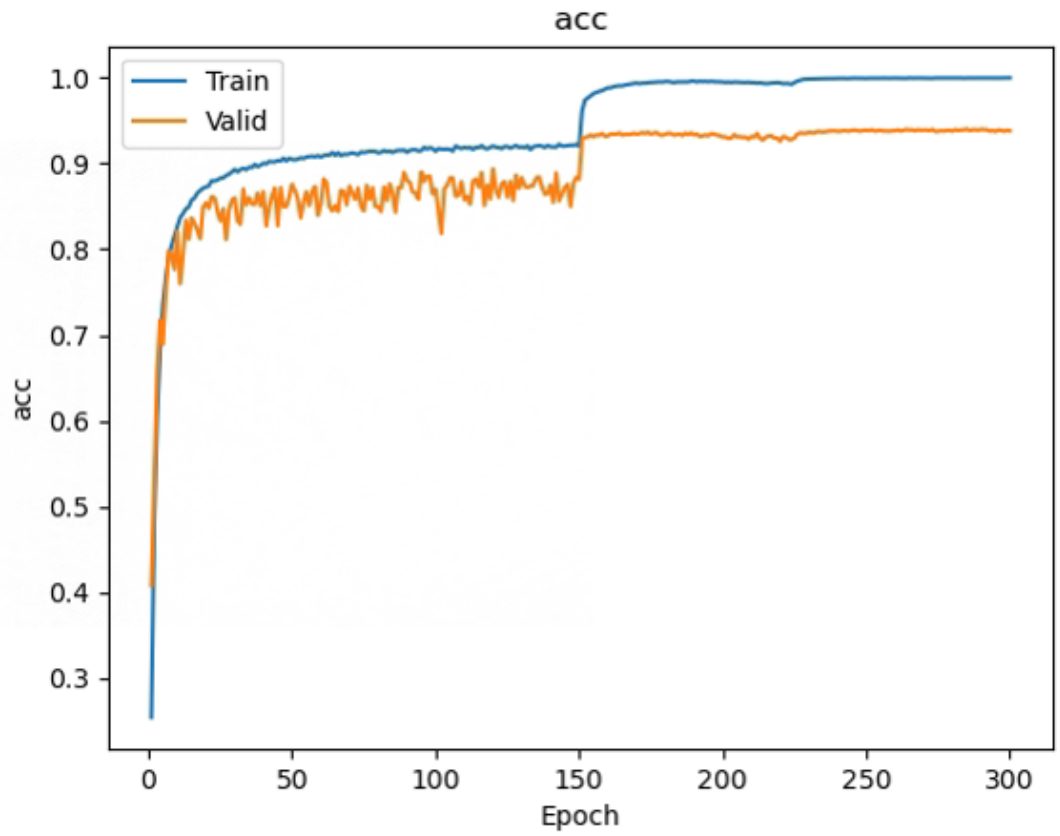


图 13: se-resnet56-acc

由上图可得，测试集中使用 se-resnet50 最终的正确率比 se-resnet20 要高，说明网络层数越深网络

模型效果越好。

## 6 总结与展望

本部分对整个文档的内容进行归纳并分析目前实现过程中的不足以及未来可进一步进行研究的方向。本次复现过程中从 SENet 选题背景依据、相关工作、方法原理等方面围绕展开，首先在选题背景依据方面可知，传统的 CNN 神经网络通常通过卷积算子，通过卷积核对特征图的局部区域进行特征融合，为了进一步提高卷积效率，从特征通道之间维度出发，通过 SE-block 块将不同特征图之间的信息融合起来，因此能够学习到图像的全局更丰富的特征，提高 CNN 提取图像特征的能力。在本篇复现论文中采用了深层 CNN 架构因为深层网络架构能更好的达到训练效果，并且还引入了注意力机制原理即借鉴了人类视觉所特有的大脑信号处理机制。人类视觉通过快速扫描全局图像，获得需要重点关注的目标区域，也就是一般所说的注意力焦点，而后对这一区域投入更多注意力资源，以获取更多所需要关注目标的细节信息，而抑制其他无用信息。除此之外本文着重介绍了 SENet 的 Squeeze 和 Excitation 两个原理模块，Squeeze 通过全局平均池化对特征图信息进行聚合，然后通过 Excitation 将特征通道之间的信息进行建模取得通道间的相关性，最后将相关性与原特征图进行加权求和得到信息聚合的特征图，以此增强特征信息。在复现过程中，在实现 SE-block 块的过程中发现 Excitation 操作的降维操作会对通道注意力的预测产生负面影响，且获取依赖关系效率低且不必要，并且在每个 conv 层都加上 SE-block 块是不必要的，因为 SE-block 块有两个全连接层，会带来大量的参数引入，因此需要在层数较浅的层加入 SE-block 块效果更好一些。虽然在 ResNet 中引入 SE-block 块中能够将性能进一步提升，但是 SE 在获取 channel attention 的模型还是太复杂了，并且引入的全连接层增加的参数加大了网络的负担，因此考虑可以在执行全局平局池化操作后直接进行特征图之间的点乘操作，而不经过 FC 层，独立学习各个特征通道之间的权值，而降维操作可以使用一个共享参数 k 个 1*1 卷积，来进行特征通道之间的权值共享，这是值得探索的一个方向。

## 7 参考文献

[1]A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015.

[3]K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In ECCV, 2016.

[4]M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In NIPS, 2015. ；