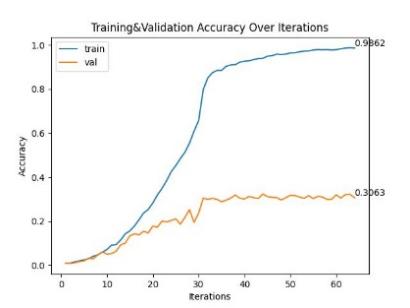
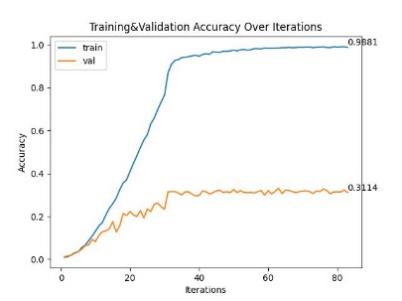
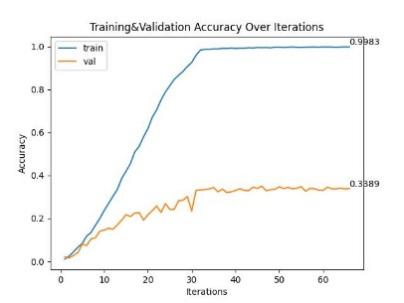
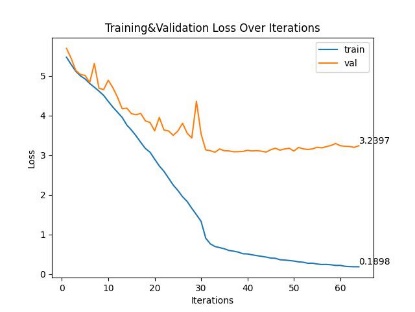
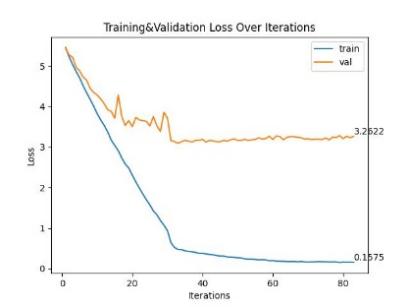
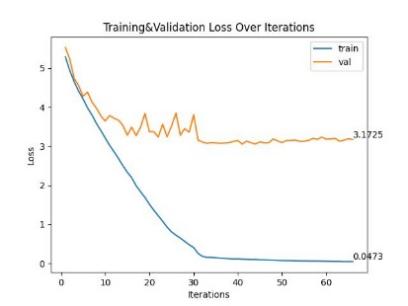
4. Experiment

没有使用迁移学习。其中early\_stopping.patience 等于20，batch\_size 等于 32，其余参数均为默认的情况下，不同模型的准确率和损失曲线如图



Accuracy，ResNet34（左），ResNet50（中），ResNet101（右）



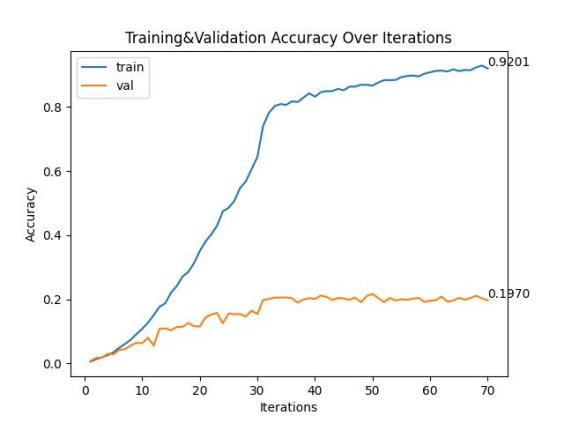
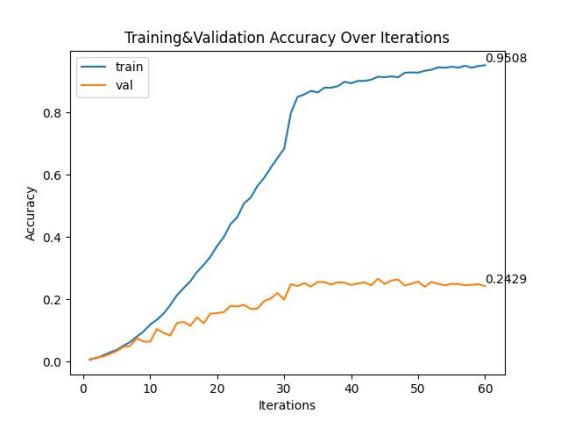
Loss，ResNet34（左），ResNet50（中），ResNet101（右）

ResNet34是34 层深的 ResNet，包含相对较少的参数和较低的复杂度，而ResNet50是50 层深的 ResNet，引入了 bottleneck 结构，使得每个残差块中的层数增加，最后ResNet101是101 层深的 ResNet，有更多的残差块和层数，更深的网络结构。

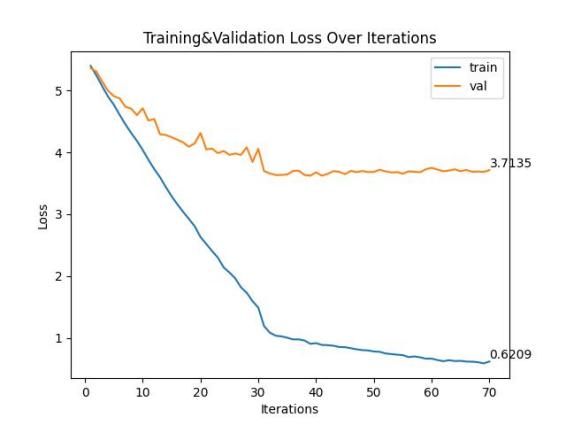
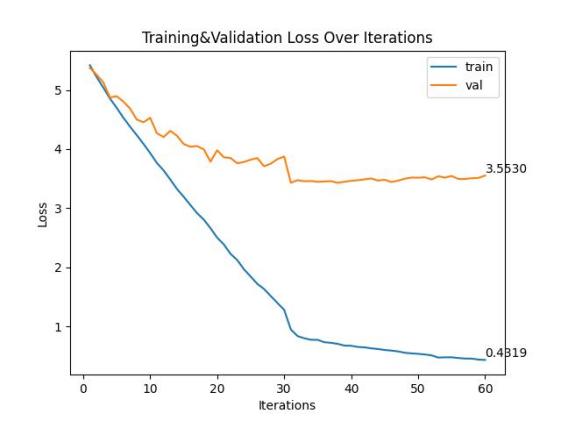
从收敛速度方面，理论上，ResNet34 会比 ResNet50 和 ResNet101 收敛得更快，因为它的网络层数较少，参数较少，计算量更小。而ResNet50 和 ResNet101 由于更深的网络结构，需要更多的迭代才能达到收敛。事实上，根据上图可以得出本次训练中三个模型的收敛速度相差不大，都是30个迭代左右收敛。但是ResNet34的曲线更陡一点。

从Accuracy方面，理论上，随着网络深度的增加，模型的表现能力增强，会更好地拟合训练数据，所以ResNet50 和 ResNet101 的精度通常会高于 ResNet34，并且ResNet101 通常可以达到最高的训练精度，因为它具有最多的参数和最大的模型容量。而事实上，由图中数据可得ResNet34的训练集Accuracy为0.9983，高于ResNet50(0.9881)和ResNet101(0.9862)，我们认为可能是因为数据量太少了或者训练时间不够长，所以比较简单的ResNet34有更好的表现。ResNet34的验证集Accuracy同样高于其余两个网络，佐证了我们的猜想。

从Loss方面，理论上，随着网络深度的增加，Loss应该会更低，因为更复杂的模型可以更好地拟合训练数据，所以ResNet101 由于其复杂性，Loss通常会比 ResNet34 和 ResNet50 更低。而事实上，Loss是ResNet101 > ResNet50 > ResNet34。ResNet34最快达到转折点并且Loss最小，我们认为原因和Accuracy中的分析相同。



batch\_size等于64（左），100（右）



batch\_size等于64（左），100（右）

使用ResNet50，其中early\_stopping.patience 等于 20，不同batch\_size的情况下的Accuracy和Loss如上图。

从收敛速度方面，理论上，较小的 batch\_size 通常会使模型的参数更新更频繁，梯度下降更快。而较大的 batch\_size 通常会使模型的参数更新较少，收敛速度可能更慢。这一点在我们的实验中体现并不明显，batch\_size在64和100都是在30个迭代左右收敛。

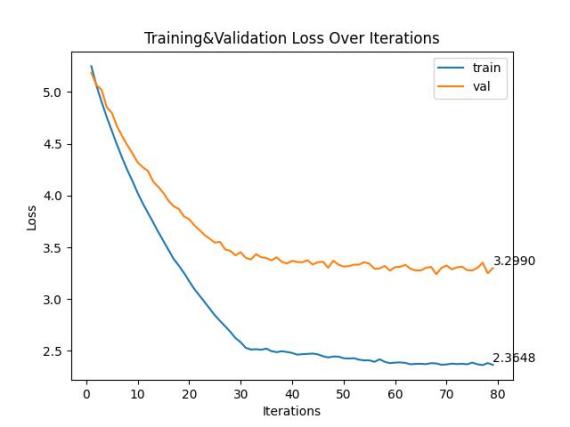
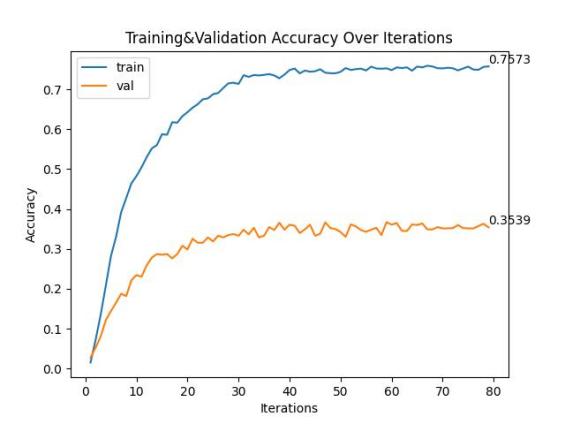
从Accuracy方面，理论上，较小的Batch\_Size由于更新频繁，模型可能会更好地适应训练数据，训练精度可能较高，并且泛化效果更好，验证精度也会更高。而较大的 batch\_size参数更新次数较少，泛化性能不如batch\_size更小的情况，精度相对较低。实际的数据表示确实是较小的batch\_size等于64的情况训练集Accuracy高于batch\_size等于100的情况（0.9508>0.9201），而验证集准确率同样是batch\_size等于64的情况更高（0.2429>0.1970）。

从Loss方面，理论上，较小的batch\_size训练损失下降更快，并且最小损失更小，验证损失同理。而较大的batch\_size训练损失下降会相对较慢，并且最小损失相对较大。这一点在实验数据中很好地体现了出来。当batch\_size较小的时候，即等于64时，训练损失和验证损失均小于batch\_size为100的情况（3.530<3.7135, 0.4319<0.6209）。

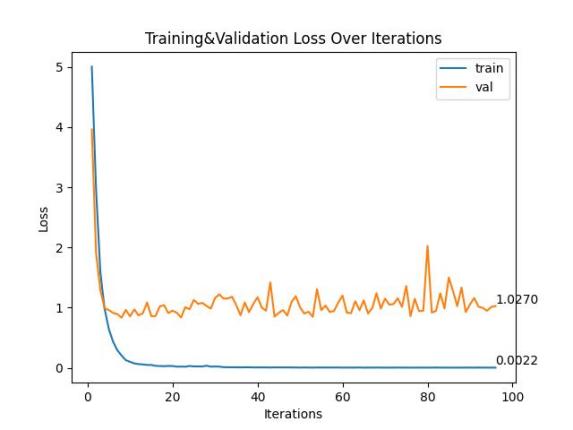
选择合适的 batch\_size 需要根据具体的任务、数据集以及计算资源进行权衡。

4. Experiment

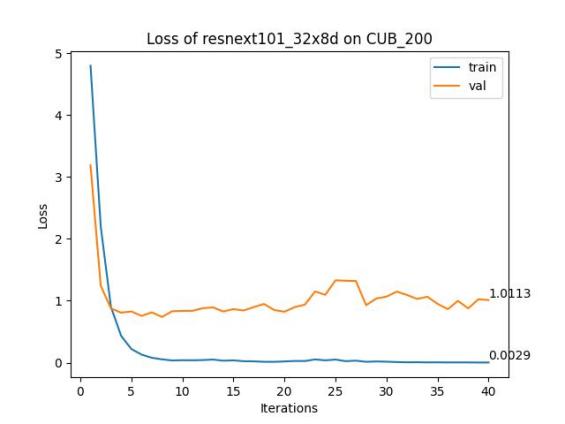
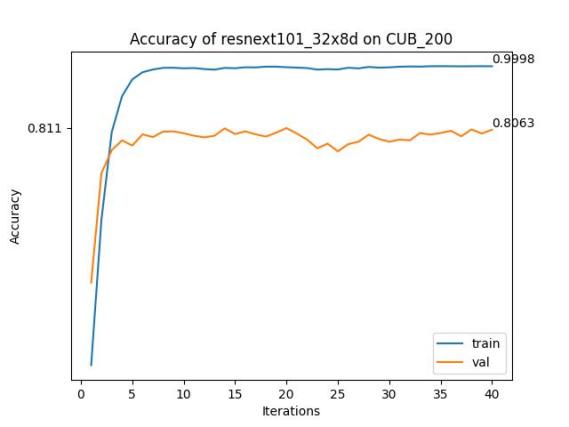
使用resnet101模型，其中lr=5e-5，batch\_size=32，earlystop.patience=20。以下是训练不同层时模型训练的结果。



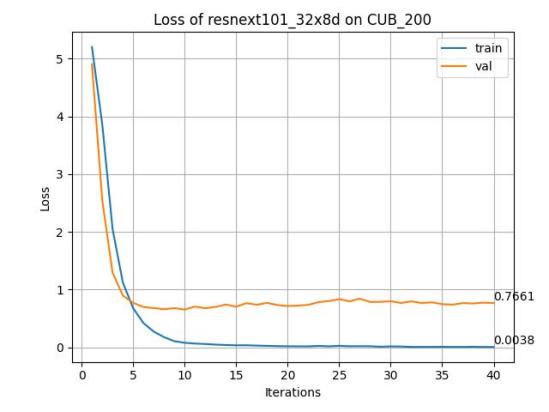
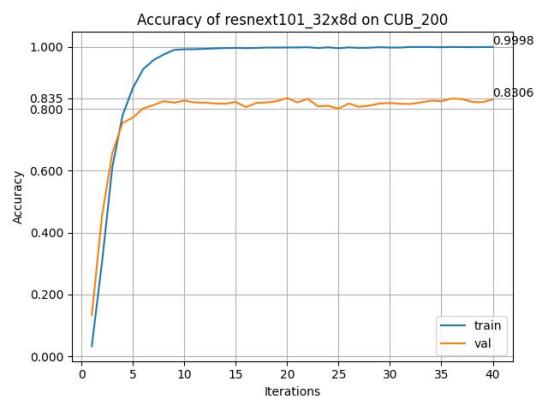
训练FC层



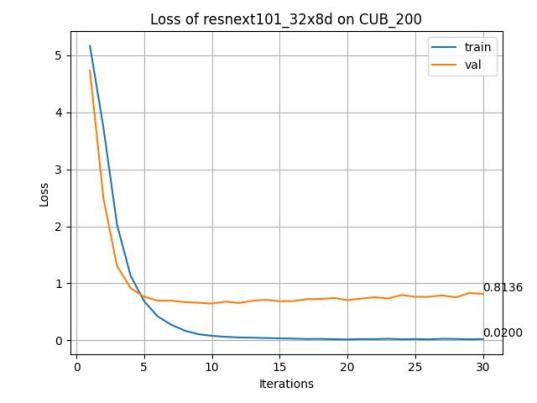
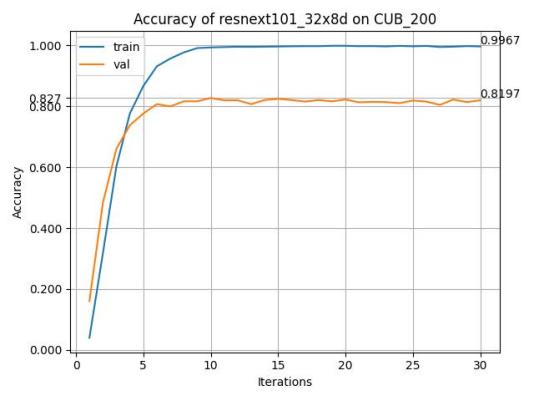
训练第四个卷积层+FC层



训练第三、第四个卷积层+FC层



训练第二，第三，第四以及FC层



训练全部层

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Accuracy | FC | 4+FC | 3+4+FC | 2+3+4+FC | ALL |
| train | 0.7573 | 0.9998 | 0.9998 | 0.9998 | 0.9967 |
| val | 0.3539 | 0.7922 | 0.8063 | 0.8306 | 0.8197 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loss | FC | 4+FC | 3+4+FC | 2+3+4+FC | ALL |
| train | 2.3648 | 0.0022 | 0.0029 | 0.0038 | 0.0200 |
| val | 3.2990 | 1.0270 | 1.0113 | 0.7661 | 0.8136 |

从实验数据层面上，先说使用迁移学习与不使用迁移学习的差别。使用迁移学习后，如果只训练FC层，冻结其他所有层，那么训练集精度不如不使用迁移学习的情况，只有0.7小于0.9；验证集精度与不使用迁移学习的情况相差无几，均为0.3左右。如果训练的层包含卷积层，则无论是训练集还是验证集精度相比于不使用迁移学习均有提升，其中训练集精度普遍到达0.99，提升不明显，但是验证集精度从0.2、0.3提升到了0.7、0.8附近，提升巨大。这是因为在数据有限的情况下，迁移学习可以帮助模型更好地泛化，因为预训练模型已经从大量数据中学到了有用的特征。从训练时间的角度，使用了迁移学习的模型普遍在10个迭代左右就达到收敛，快于不使用迁移学习的情况（大约30个迭代），这是因为预训练模型已经学到了通用特征，只需要在目标任务上进行微调，因此大大减少了训练时间。

从训练不同层的角度出发。显然仅训练FC层时模型的性能最差，无论是训练集还是验证集，无论是精度还是损失；而训练2+3+4+FC层时模型的性能最好，验证集精度达到了不错的0.8306。这是因为预训练模型学到的特征主要针对源任务，这些特征虽然在某种程度上是通用的，但在目标任务上可能不够专用。仅仅训练全连接层意味着模型只能在已经提取好的特征上进行学习，如果这些特征在目标任务上不是最优的，那么全连接层的表现也会受到限制。而为什么随着训练的卷积层增多，验证集精度在上升，但是当训练全部层时验证集精度反而略有下降。这可能是因为预训练模型中的权重已经在大量数据上进行了优化，这些权重在很大程度上捕捉了数据的通用特征，对整个模型进行微调可能会破坏这些权重，使模型在新任务上反而表现不佳。我们认为这更可能是因为所有层都在调整，模型可能会过度拟合目标任务的数据，失去预训练权重中捕捉到的通用特征，导致泛化性能下降。

总的来说，迁移学习利用源任务中学到的知识来帮助目标任务的学习，从而提高目标任务的模型性能并减少训练时间，特别适用于在数据有限或训练时间受限的情况。在本实验中迁移学习的作用十分显著。