SMOTE 算法：合成少数过采样

根据已有的样本合成新的样本，实现对少数样本的过采样。它会选择某个已有样本，找到离这个样本最近的 k 个样本，随机选择一个样本，然后在这两个样本之间的某点随机生成新的样本。

该方法的局限性：当少数样本点比较稀疏（样本之间相距较远），可能会生成噪声样本。

算法伪代码：

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100) * T$ synthetic minority class samples
(∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
if $N < 100$
  then Randomize the T minority class samples
    $T = (N/100) * T$
    $N = 100$
endif
$N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
k = Number of nearest neighbors
$numattrs$ = Number of attributes
$Sample[\,][\,]$: array for original minority class samples
$newindex$: keeps a count of number of synthetic samples generated, initialized to 0
$Synthetic[\,][\,]$: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
for $i \leftarrow 1$ **to** T
    Compute k nearest neighbors for i, and save the indices in the $nnarray$
    Populate(N, i, $nnarray$)
endfor

Populate(N, i, $nnarray$) (∗ *Function to generate the synthetic samples.* ∗)
while $N \neq 0$
    Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
    for $attr \leftarrow 1$ **to** $numattrs$
      Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
      Compute: gap = random number between 0 and 1
      $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
    endfor
    $newindex{+}{+}$
    $N = N - 1$
endwhile
return (∗ *End of Populate.* ∗)
End of Pseudo-Code.

具体实现可以看 smote_implementation.py。imblearn 库中内置了 SMOTE 算法，使用方法：

```
from imblearn.over_sampling import SMOTE
# 使用SMOTE过采样处理不平衡的数据集
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y) # type: ignore
```

其中 random_state 为随机数种子，X 和 y 为初识数据集的特征和标签，X_resampled 和 y_resampled 是合成后的特征和标签。

一般而言，对多数样本进行欠采样的效果要比对少数样本过采样的效果更好，所以 SMOTE 算法通常会和欠采样结合，比如上一篇论文 “Hybrid_CNN-LSTM_With_Attention_Mechanism_for_Robust_Credit_Card_Fraud_Detection” 中使用 SMOTE 算法后多数样本变少了。

尝试使用 SMOTE 算法：

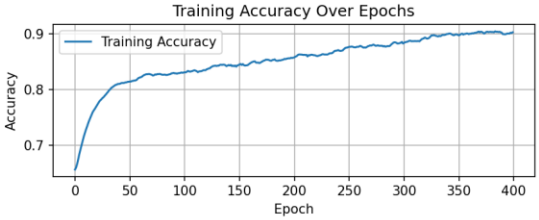
具体代码参照论文下 main.py。使用的数据集为 SMOTE 论文中的第一个数据集 The Pima Indian Diabetes，该数据集共有 768 个样本，每个样本有 8 个特征和 1 个标签，标签只会为 0 或 1，属于经典的二分类数据集，但是这个数据集中的标签比例是不平衡的，标签为 0 个样本有 500 个，标签为 1 的样本有 268 个：

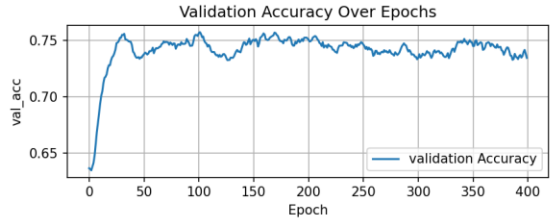
```python
import pandas as pd
file = pd.read_csv("./The Pima Indian Diabetes.csv")

outcome = file["Outcome"]
print("count 0:", outcome.to_list().count(0))
print("count 1:", outcome.to_list().count(1))
```

```
count 0: 500
count 1: 268
```

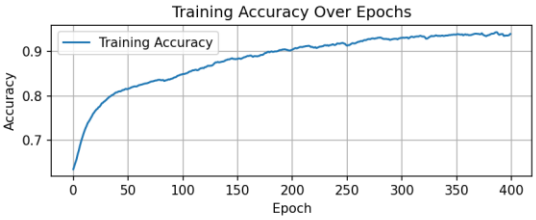
首先正常处理数据集，这里我只拆分成了两部分，所以只有训练集和验证集，没有留测试集，验证集不参与训练。

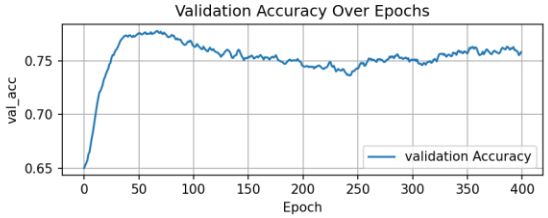
不使用 SMOTE 算法，神经网络使用线性层，最后可以得到拟合准确率：





使用 SMOTE 算法后的拟合结果：





上面的可视化图均为使用数据平滑后的结果，使用 SMOTE 算法时在验证集上可以达到近 80%的准确率，而不使用 SMOTE 算法准确率只能达到 75%，相较而言 SMOTE 算法对模型有一定的提升。（多次测试发现不使用 SMOTE 算法偶尔也可以达到近 80%的准确率，这个峰值很不稳定）。