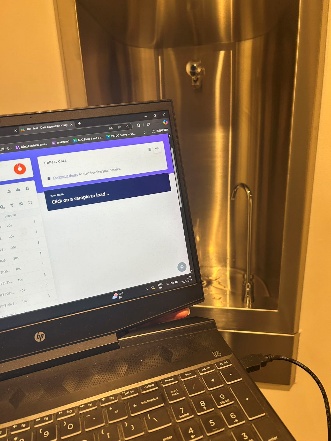
Week1

数据集收集。

Week2

使用test 3已有数据集。

tes6.49已处理是我个人收集的数据集。

# 关于test 3的尝试：

根据tutorial的步骤制作的模型并不准确，错误识别为**faucet概率很高。我已经尝试了默认设置的classification，接下来将全部使用transfer learning。**

**但是教程中的epoch为30，我认为不对，我将尝试找到最佳的epoch.**

1/100

42/42 - 12s - loss: 0.5189 - accuracy: 0.9114 - val\_loss: 0.3715 - val\_accuracy: 0.9397 - 12s/ - 295ms/step

2/100

42/42 - 8s - loss: 0.3097 - accuracy: 0.9389 - val\_loss: 0.2520 - val\_accuracy: 0.9464 - 8s/ - 187ms/step

3/100

42/42 - 8s - loss: 0.2318 - accuracy: 0.9419 - val\_loss: 0.2006 - val\_accuracy: 0.9487 - 8s/ - 179ms/step

4/100

42/42 - 8s - loss: 0.1942 - accuracy: 0.9412 - val\_loss: 0.1736 - val\_accuracy: 0.9501 - 8s/ - 181ms/step

5/100

loss: 0.1711 - accuracy: 0.9479 - val\_loss: 0.1563 - val\_accuracy: 0.9509 - 7s/ - 172ms/step

6/100

loss: 0.1556 - accuracy: 0.9516 - val\_loss: 0.1434 - val\_accuracy: 0.9531 - 7s/ - 171ms/step

7/100

loss: 0.1433 - accuracy: 0.9576 - val\_loss: 0.1352 - val\_accuracy: 0.9546 - 7s/ - 174ms/step

8/100

loss: 0.1327 - accuracy: 0.9583 - val\_loss: 0.1276 - val\_accuracy: 0.9568 - 7s/ - 175ms/step

9/100

loss: 0.1279 - accuracy: 0.9598 - val\_loss: 0.1217 - val\_accuracy: 0.9576 - 7s/ - 175ms/step

10/100

loss: 0.1207 - accuracy: 0.9590 - val\_loss: 0.1171 - val\_accuracy: 0.9568 - 7s/ - 173ms/step

11/100

loss: 0.1167 - accuracy: 0.9620 - val\_loss: 0.1131 - val\_accuracy: 0.9576 - 7s/ - 166ms/step

12/100

loss: 0.1113 - accuracy: 0.9643 - val\_loss: 0.1098 - val\_accuracy: 0.9591 - 7s/ - 173ms/step

13/100

loss: 0.1067 - accuracy: 0.9643 - val\_loss: 0.1068 - val\_accuracy: 0.9591 - 7s/ - 174ms/step

14/100

loss: 0.1045 - accuracy: 0.9650 - val\_loss: 0.1042 - val\_accuracy: 0.9576 - 7s/ - 173ms/step

15/100

42/42 - 8s - loss: 0.0987 - accuracy: 0.9672 - val\_loss: 0.1017 - val\_accuracy: 0.9583 - 8s/ - 181ms/step

16/100

loss: 0.0971 - accuracy: 0.9695 - val\_loss: 0.0991 - val\_accuracy: 0.9583 - 7s/ - 175ms/step

17/100

42/42 - 8s - loss: 0.0924 - accuracy: 0.9732 - val\_loss: 0.0979 - val\_accuracy: 0.9576 - 8s/ - 179ms/step

18/100

loss: 0.0909 - accuracy: 0.9710 - val\_loss: 0.0961 - val\_accuracy: 0.9576 - 7s/ - 175ms/step

19/100

42/42 - 8s - loss: 0.0877 - accuracy: 0.9710 - val\_loss: 0.0940 - val\_accuracy: 0.9583 - 8s/ - 179ms/step

20/100

loss: 0.0854 - accuracy: 0.9724 - val\_loss: 0.0925 - val\_accuracy: 0.9606 - 7s/ - 178ms/step

21/100

loss: 0.0841 - accuracy: 0.9739 - val\_loss: 0.0913 - val\_accuracy: 0.9606 - 7s/ - 175ms/step

22/100

42/42 - 9s - loss: 0.0814 - accuracy: 0.9732 - val\_loss: 0.0903 - val\_accuracy: 0.9621 - 9s/ - 209ms/step

23/100

loss: 0.0787 - accuracy: 0.9762 - val\_loss: 0.0894 - val\_accuracy: 0.9643 - 7s/ - 178ms/step

24/100

23/100

loss: 0.0787 - accuracy: 0.9762 - val\_loss: 0.0894 - val\_accuracy: 0.9643 - 7s/ - 178ms/step

loss: 0.0776 - accuracy: 0.9724 - val\_loss: 0.0877 - val\_accuracy: 0.9621 - 7s/ - 176ms/step

25/100

loss: 0.0756 - accuracy: 0.9792 - val\_loss: 0.0867 - val\_accuracy: 0.9635 - 7s/ - 172ms/step

26/100

loss: 0.0740 - accuracy: 0.9747 - val\_loss: 0.0858 - val\_accuracy: 0.9650 - 7s/ - 169ms/step

27/100

loss: 0.0745 - accuracy: 0.9762 - val\_loss: 0.0852 - val\_accuracy: 0.9680 - 7s/ - 166ms/step

28/100

loss: 0.0715 - accuracy: 0.9784 - val\_loss: 0.0841 - val\_accuracy: 0.9673 - 7s/ - 167ms/step

29/100

loss: 0.0690 - accuracy: 0.9777 - val\_loss: 0.0834 - val\_accuracy: 0.9688 - 7s/ - 166ms/step

30/100

loss: 0.0687 - accuracy: 0.9762 - val\_loss: 0.0823 - val\_accuracy: 0.9665 - 7s/ - 164ms/step

31/100

loss: 0.0681 - accuracy: 0.9769 - val\_loss: 0.0817 - val\_accuracy: 0.9688 - 7s/ - 164ms/step

32/100

loss: 0.0672 - accuracy: 0.9806 - val\_loss: 0.0814 - val\_accuracy: 0.9695 - 7s/ - 166ms/step

33/100

loss: 0.0655 - accuracy: 0.9792 - val\_loss: 0.0809 - val\_accuracy: 0.9695 - 7s/ - 170ms/step

34/100

loss: 0.0651 - accuracy: 0.9799 - val\_loss: 0.0799 - val\_accuracy: 0.9688 - 7s/ - 171ms/step

35/100

loss: 0.0631 - accuracy: 0.9777 - val\_loss: 0.0796 - val\_accuracy: 0.9695 - 7s/ - 174ms/step

36/100

loss: 0.0627 - accuracy: 0.9814 - val\_loss: 0.0791 - val\_accuracy: 0.9695 - 7s/ - 172ms/step

37/100

42/42 - 9s - loss: 0.0623 - accuracy: 0.9821 - val\_loss: 0.0791 - val\_accuracy: 0.9725 - 9s/ - 203ms/step

38/100

loss: 0.0604 - accuracy: 0.9851 - val\_loss: 0.0780 - val\_accuracy: 0.9702 - 7s/ - 164ms/step

39/100

loss: 0.0576 - accuracy: 0.9806 - val\_loss: 0.0778 - val\_accuracy: 0.9725 - 7s/ - 169ms/step

40/100

loss: 0.0561 - accuracy: 0.9851 - val\_loss: 0.0774 - val\_accuracy: 0.9732 - 7s/ - 172ms/step

41/100

loss: 0.0575 - accuracy: 0.9829 - val\_loss: 0.0765 - val\_accuracy: 0.9710 - 7s/ - 177ms/step

42/100

loss: 0.0557 - accuracy: 0.9859 - val\_loss: 0.0763 - val\_accuracy: 0.9717 - 7s/ - 172ms/step

43/100

loss: 0.0524 - accuracy: 0.9859 - val\_loss: 0.0758 - val\_accuracy: 0.9725 - 7s/ - 172ms/step

44/100

loss: 0.0530 - accuracy: 0.9866 - val\_loss: 0.0753 - val\_accuracy: 0.9725 - 7s/ - 176ms/step

45/100

loss: 0.0528 - accuracy: 0.9873 - val\_loss: 0.0754 - val\_accuracy: 0.9740 - 7s/ - 169ms/step

46/100

loss: 0.0533 - accuracy: 0.9844 - val\_loss: 0.0747 - val\_accuracy: 0.9740 - 7s/ - 176ms/step

47/100

loss: 0.0508 - accuracy: 0.9881 - val\_loss: 0.0745 - val\_accuracy: 0.9740 - 7s/ - 173ms/step

48/100

loss: 0.0517 - accuracy: 0.9859 - val\_loss: 0.0741 - val\_accuracy: 0.9740 - 7s/ - 173ms/step

49/100

loss: 0.0511 - accuracy: 0.9851 - val\_loss: 0.0742 - val\_accuracy: 0.9732 - 7s/ - 168ms/step

50/100

loss: 0.0500 - accuracy: 0.9859 - val\_loss: 0.0738 - val\_accuracy: 0.9732 - 7s/ - 176ms/step

51/100

loss: 0.0477 - accuracy: 0.9881 - val\_loss: 0.0732 - val\_accuracy: 0.9732 - 7s/ - 178ms/step

52/100

loss: 0.0491 - accuracy: 0.9866 - val\_loss: 0.0728 - val\_accuracy: 0.9732 - 7s/ - 175ms/step

53/100

loss: 0.0479 - accuracy: 0.9866 - val\_loss: 0.0727 - val\_accuracy: 0.9740 - 7s/ - 173ms/step

54/100

loss: 0.0475 - accuracy: 0.9896 - val\_loss: 0.0724 - val\_accuracy: 0.9740 - 7s/ - 174ms/step

55/100

loss: 0.0448 - accuracy: 0.9918 - val\_loss: 0.0722 - val\_accuracy: 0.9740 - 7s/ - 170ms/step

56/100

loss: 0.0461 - accuracy: 0.9881 - val\_loss: 0.0721 - val\_accuracy: 0.9740 - 7s/ - 173ms/step

57/100

loss: 0.0454 - accuracy: 0.9896 - val\_loss: 0.0715 - val\_accuracy: 0.9725 - 7s/ - 168ms/step

58/100

loss: 0.0452 - accuracy: 0.9888 - val\_loss: 0.0717 - val\_accuracy: 0.9740 - 7s/ - 163ms/step

59/100

loss: 0.0437 - accuracy: 0.9881 - val\_loss: 0.0716 - val\_accuracy: 0.9740 - 7s/ - 164ms/step

60/100

loss: 0.0441 - accuracy: 0.9896 - val\_loss: 0.0718 - val\_accuracy: 0.9725 - 7s/ - 162ms/step

61/100

loss: 0.0430 - accuracy: 0.9903 - val\_loss: 0.0713 - val\_accuracy: 0.9732 - 7s/ - 172ms/step

62/100

loss: 0.0429 - accuracy: 0.9911 - val\_loss: 0.0710 - val\_accuracy: 0.9732 - 7s/ - 172ms/step

63/100

loss: 0.0433 - accuracy: 0.9896 - val\_loss: 0.0707 - val\_accuracy: 0.9740 - 7s/ - 169ms/step

64/100

loss: 0.0413 - accuracy: 0.9896 - val\_loss: 0.0708 - val\_accuracy: 0.9732 - 7s/ - 164ms/step

65/100

loss: 0.0424 - accuracy: 0.9896 - val\_loss: 0.0708 - val\_accuracy: 0.9732 - 7s/ - 164ms/step

66/100

loss: 0.0407 - accuracy: 0.9918 - val\_loss: 0.0705 - val\_accuracy: 0.9725 - 7s/ - 168ms/step

67/100

loss: 0.0407 - accuracy: 0.9918 - val\_loss: 0.0705 - val\_accuracy: 0.9725 - 7s/ - 168ms/step

67/100

loss: 0.0409 - accuracy: 0.9911 - val\_loss: 0.0703 - val\_accuracy: 0.9725 - 7s/ - 173ms/step

68/100

loss: 0.0402 - accuracy: 0.9918 - val\_loss: 0.0704 - val\_accuracy: 0.9732 - 7s/ - 170ms/step

69/100

loss: 0.0410 - accuracy: 0.9896 - val\_loss: 0.0703 - val\_accuracy: 0.9717 - 7s/ - 167ms/step

70/100

loss: 0.0382 - accuracy: 0.9918 - val\_loss: 0.0706 - val\_accuracy: 0.9740 - 7s/ - 165ms/step

71/100

loss: 0.0375 - accuracy: 0.9918 - val\_loss: 0.0703 - val\_accuracy: 0.9732 - 7s/ - 173ms/step

72/100

loss: 0.0369 - accuracy: 0.9933 - val\_loss: 0.0701 - val\_accuracy: 0.9740 - 7s/ - 177ms/step

73/100

42/42 - 8s - loss: 0.0383 - accuracy: 0.9911 - val\_loss: 0.0695 - val\_accuracy: 0.9732 - 8s/ - 182ms/step

74/100

loss: 0.0390 - accuracy: 0.9918 - val\_loss: 0.0694 - val\_accuracy: 0.9740 - 7s/ - 171ms/step

75/100

loss: 0.0391 - accuracy: 0.9911 - val\_loss: 0.0690 - val\_accuracy: 0.9732 - 7s/ - 170ms/step

76/100

loss: 0.0363 - accuracy: 0.9903 - val\_loss: 0.0690 - val\_accuracy: 0.9732 - 7s/ - 175ms/step

77/100

loss: 0.0369 - accuracy: 0.9918 - val\_loss: 0.0692 - val\_accuracy: 0.9747 - 7s/ - 170ms/step

78/100

loss: 0.0355 - accuracy: 0.9896 - val\_loss: 0.0691 - val\_accuracy: 0.9747 - 7s/ - 171ms/step

79/100

loss: 0.0345 - accuracy: 0.9948 - val\_loss: 0.0687 - val\_accuracy: 0.9732 - 7s/ - 174ms/step

80/100

loss: 0.0348 - accuracy: 0.9926 - val\_loss: 0.0686 - val\_accuracy: 0.9725 - 7s/ - 173ms/step

81/100

loss: 0.0383 - accuracy: 0.9888 - val\_loss: 0.0688 - val\_accuracy: 0.9754 - 7s/ - 169ms/step

82/100

loss: 0.0331 - accuracy: 0.9933 - val\_loss: 0.0687 - val\_accuracy: 0.9725 - 7s/ - 171ms/step

83/100

loss: 0.0336 - accuracy: 0.9940 - val\_loss: 0.0686 - val\_accuracy: 0.9740 - 7s/ - 171ms/step

84/100

loss: 0.0331 - accuracy: 0.9948 - val\_loss: 0.0684 - val\_accuracy: 0.9732 - 7s/ - 176ms/step

85/100

loss: 0.0335 - accuracy: 0.9940 - val\_loss: 0.0683 - val\_accuracy: 0.9725 - 7s/ - 174ms/step

86/100

loss: 0.0312 - accuracy: 0.9940 - val\_loss: 0.0684 - val\_accuracy: 0.9725 - 7s/ - 169ms/step

87/100

loss: 0.0321 - accuracy: 0.9926 - val\_loss: 0.0685 - val\_accuracy: 0.9725 - 7s/ - 171ms/step

88/100

42/42 - 9s - loss: 0.0347 - accuracy: 0.9933 - val\_loss: 0.0685 - val\_accuracy: 0.9740 - 9s/ - 206ms/step

89/100

loss: 0.0313 - accuracy: 0.9940 - val\_loss: 0.0684 - val\_accuracy: 0.9747 - 7s/ - 166ms/step

90/100

loss: 0.0323 - accuracy: 0.9933 - val\_loss: 0.0680 - val\_accuracy: 0.9732 - 7s/ - 170ms/step

91/100

loss: 0.0319 - accuracy: 0.9933 - val\_loss: 0.0681 - val\_accuracy: 0.9725 - 7s/ - 165ms/step

92/100

loss: 0.0307 - accuracy: 0.9933 - val\_loss: 0.0684 - val\_accuracy: 0.9717 - 7s/ - 171ms/step

93/100

loss: 0.0309 - accuracy: 0.9940 - val\_loss: 0.0682 - val\_accuracy: 0.9717 - 7s/ - 171ms/step

94/100

loss: 0.0291 - accuracy: 0.9918 - val\_loss: 0.0682 - val\_accuracy: 0.9717 - 7s/ - 170ms/step

95/100

loss: 0.0290 - accuracy: 0.9948 - val\_loss: 0.0681 - val\_accuracy: 0.9717 - 7s/ - 174ms/step

96/100

loss: 0.0292 - accuracy: 0.9940 - val\_loss: 0.0687 - val\_accuracy: 0.9740 - 7s/ - 171ms/step

97/100

loss: 0.0303 - accuracy: 0.9948 - val\_loss: 0.0682 - val\_accuracy: 0.9725 - 7s/ - 174ms/step

98/100

loss: 0.0309 - accuracy: 0.9926 - val\_loss: 0.0681 - val\_accuracy: 0.9710 - 7s/ - 171ms/step

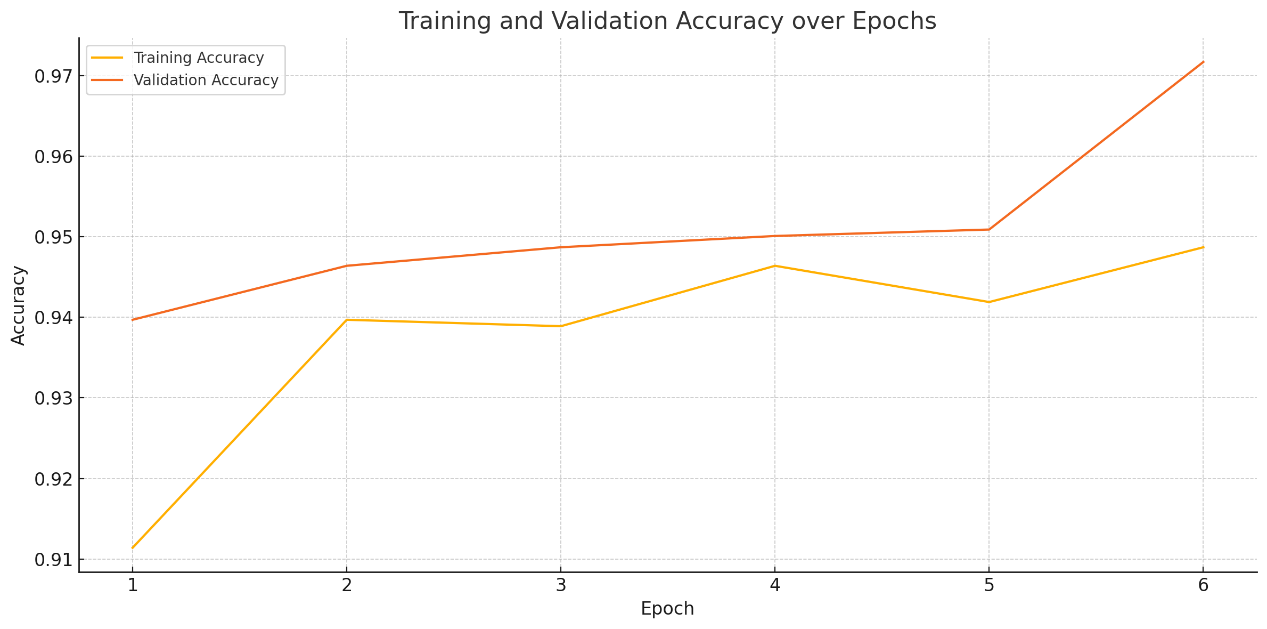
99/100

loss: 0.0283 - accuracy: 0.9933 - val\_loss: 0.0682 - val\_accuracy: 0.9717 - 7s/ - 173ms/step

100/100

loss: 0.0262 - accuracy: 0.9970 - val\_loss: 0.0684 - val\_accuracy: 0.9717 - 7s/ - 171ms/step

Finished training



55 loss: 0.0448 - accuracy: 0.9918 - val\_loss: 0.0722 - val\_accuracy: 0.9740 - 7s/ - 170ms/step

72/100

loss: 0.0369 - accuracy: 0.9933 - val\_loss: 0.0701 - val\_accuracy: 0.9740 - 7s/ - 177ms/step

79/100

loss: 0.0345 - accuracy: 0.9948 - val\_loss: 0.0687 - val\_accuracy: 0.9732 - 7s/ - 174ms/step

89/100

loss: 0.0313 - accuracy: 0.9940 - val\_loss: 0.0684 - val\_accuracy: 0.9747 - 7s/ - 166ms/step

95/100

loss: 0.0290 - accuracy: 0.9948 - val\_loss: 0.0681 - val\_accuracy: 0.9717 - 7s/ - 174ms/step

97/100

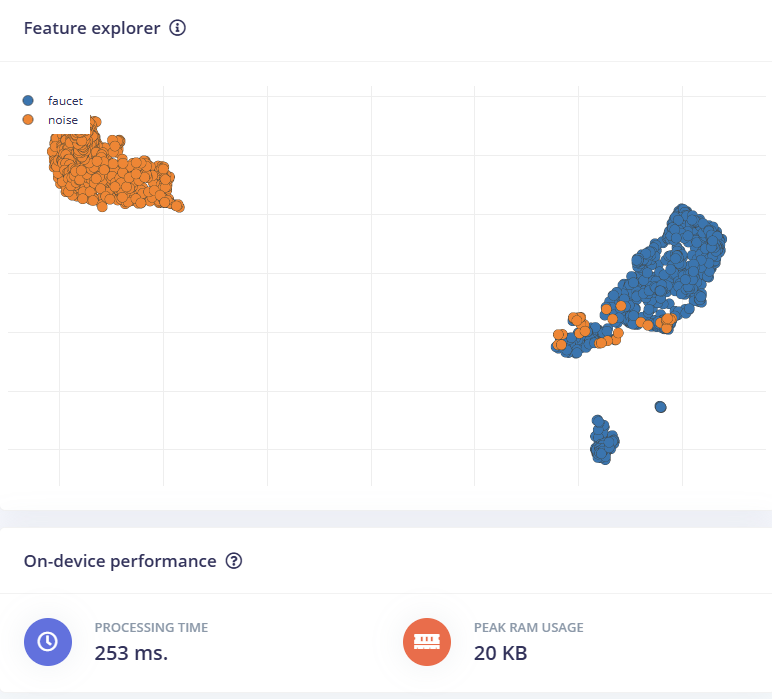
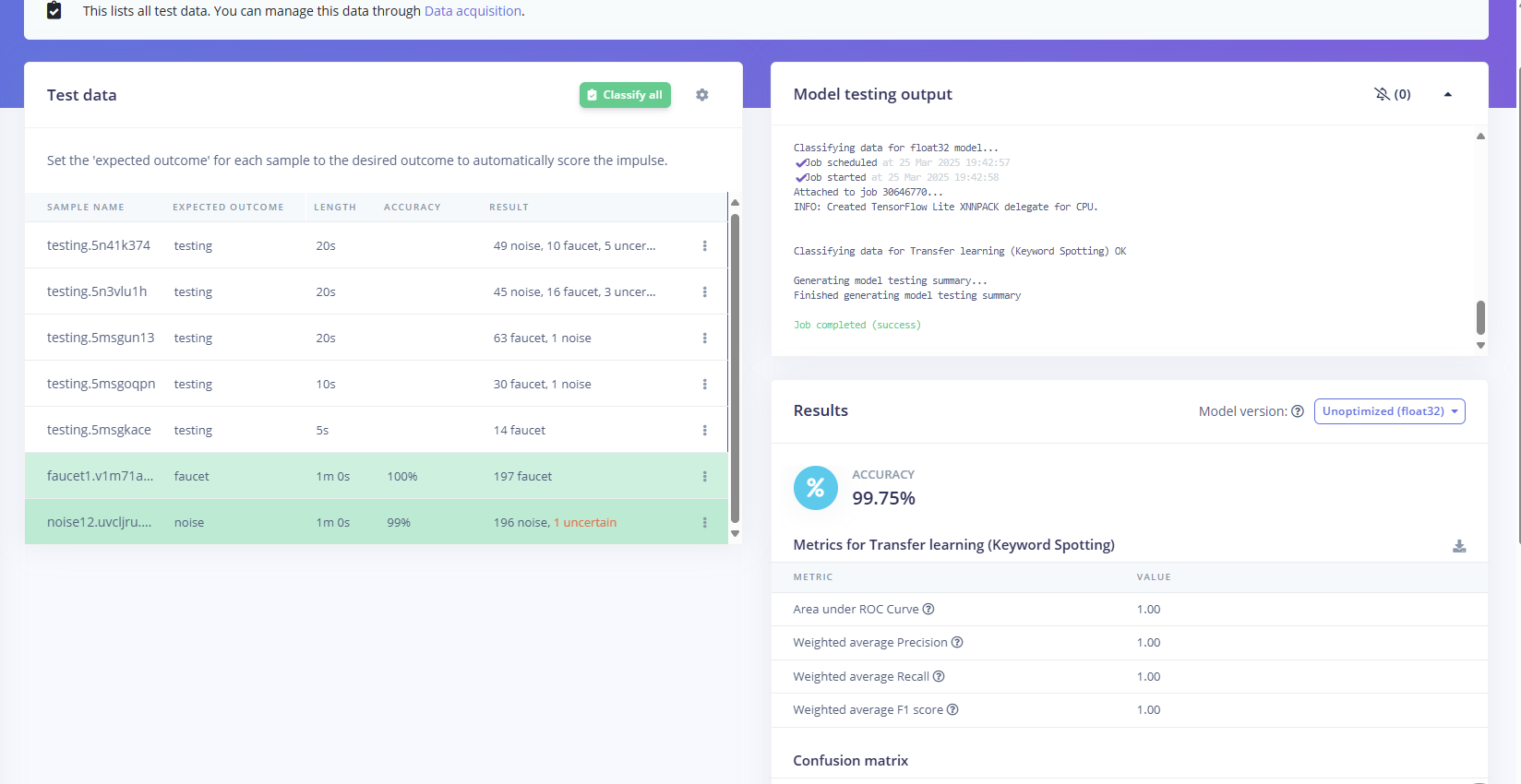
loss: 0.0303 - accuracy: 0.9948 - val\_loss: 0.0682 - val\_accuracy: 0.9725 - 7s/ - 174ms/step

100/100

loss: 0.0262 - accuracy: 0.9970 - val\_loss: 0.0684 - val\_accuracy: 0.9717 - 7s/ - 171ms/step

尽管accuracy还在提高，但val accuracy从epoch89后再没提升过所有我认为最佳的epoch是89，考虑到性能和时长，我将之后的epoch都设置为80

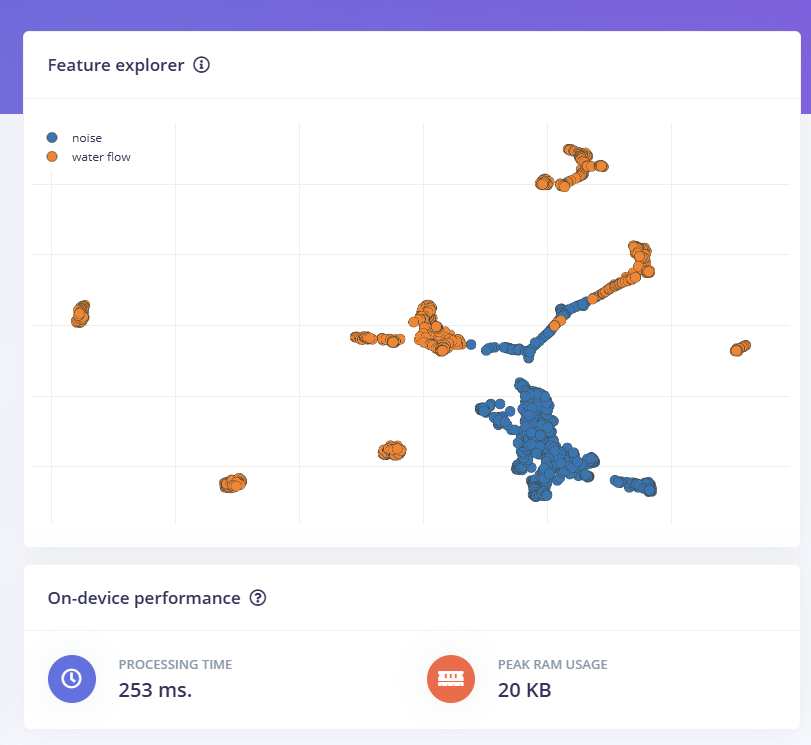
即便如此，输出的模型还是完全不准确，上图中低于0.5的点都是误判。可惜的是我丢失了epoch 为30的那个模型，所以我无法进行对比。但我目前认为是数据集的问题。

改为80epoch也没优化。  
毕竟无论是噪音还是水流声的特征点都过于集中，鲁棒性必然不好。

Modeling test的测试结果是同一数据集中的测试数据结果优秀，但是我自己录制的效果却很差。也可能是validationset太大了，我打了50，默认是20.我在下一版中改了回来，试试。

还是不行。几乎肯定是数据集的问题了。

# tes6.49已处理的尝试：

  
两种类别的数据点重叠非常严重

没有明显的边界可以把 water flow 和 noise 区分开

有很多 orange 点（水流声）“插”在蓝色区域（背景噪声）里，反之亦然

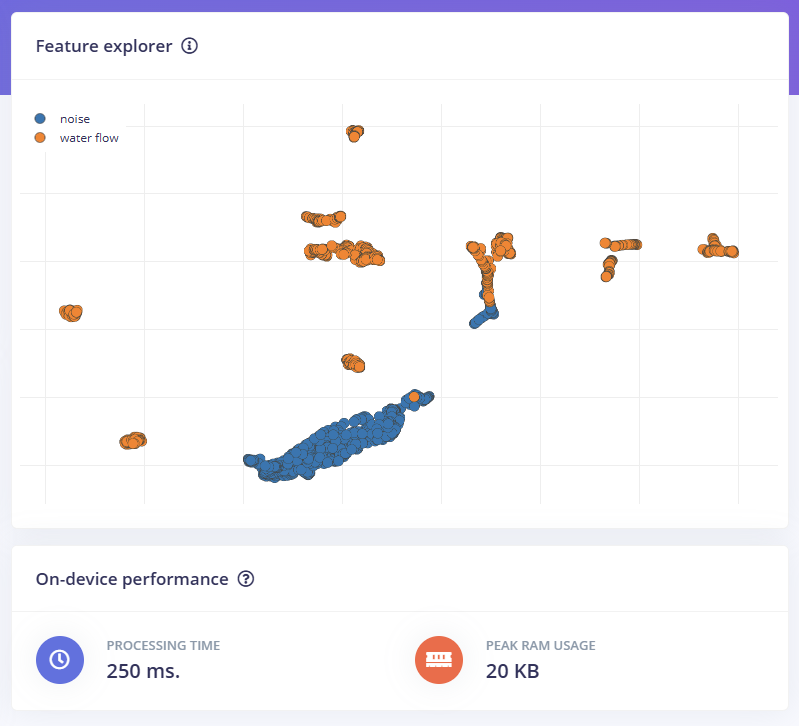
|  |  |
| --- | --- |
| Frame length | 从 0.02 → 0.03 或 0.04（增加每帧时间） |

|  |  |
| --- | --- |
| Filter number | 从 40 → 60（更细的频率划分） |

|  |  |
| --- | --- |
| FFT length | 从 256 → 512（更高频率分辨率） |

|  |  |
| --- | --- |
| Low frequency | 试试从 300Hz 改为 100Hz（包含低频水声） |

|  |  |
| --- | --- |
| High frequency | 确保等于采样率的一半，比如 8000Hz |



🟠 water flow 和 🔵 noise 现在已经开始分出一些明显的聚类块

蓝色的 noise 聚集得很紧密，说明特征相对统一（模型容易学）

橙色的 water flow 分散得稍微多一些，但也有清晰的多个子聚类

两类之间的“混色区域”明显减少

改进：

1尝试把水流声音按“流速”分为几类（例如 fast flow / slow flow），这样可能帮助模型分辨更细的特征，也可能降低类内变异。

2现在的 noise 表现非常好，但你可以再录一些更“复杂”的环境音（人说话、厕所冲水、电器运转），测试模型抗干扰能力。

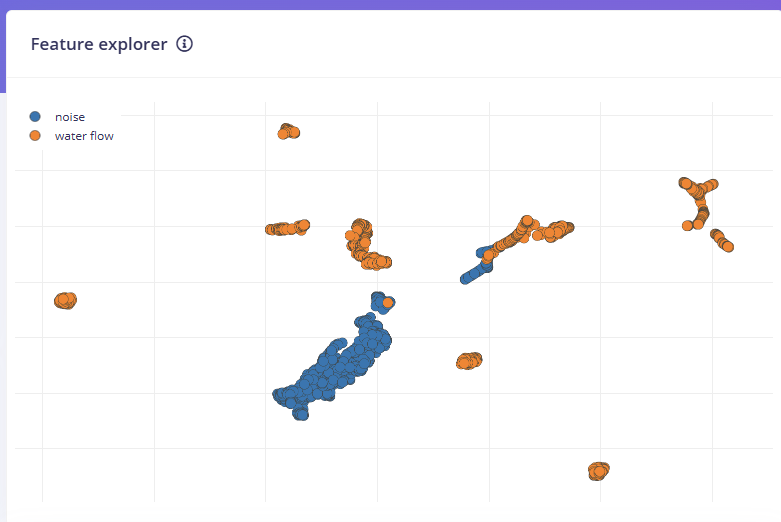
3

| **参数** | **可以尝试** |
| --- | --- |

|  |  |
| --- | --- |
| Filter number | 40 → 60 |

|  |  |
| --- | --- |
| FFT length | 256 → 512 |

|  |  |
| --- | --- |
| Frame length | 0.02 → 0.03 |



我也想在数据集中加入使用花洒的数据，但是使用花洒会有大量水雾，我担心它会破坏电路，在我完成防水设计前，我可能无法取得这部分数据。

我仍在继续优化数据集，目前家里的厨房还未出现在数据集中，可以作为testing set 的录制来使用。

数据集优化完成，第一次使用classification来训练，配置如下Input

→ Reshape (60 columns)

→ Conv1D (8 filters)

→ Dropout

→ Conv1D (16 filters)

→ Dropout

→ Conv1D (32 filters)

→ Dropout (可选)

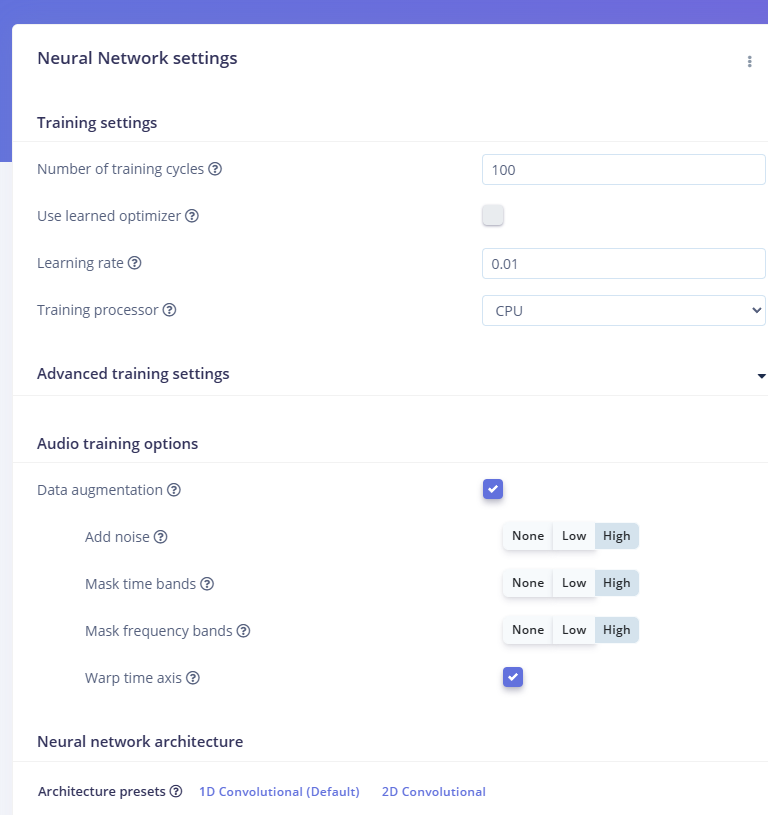
→ Flatten

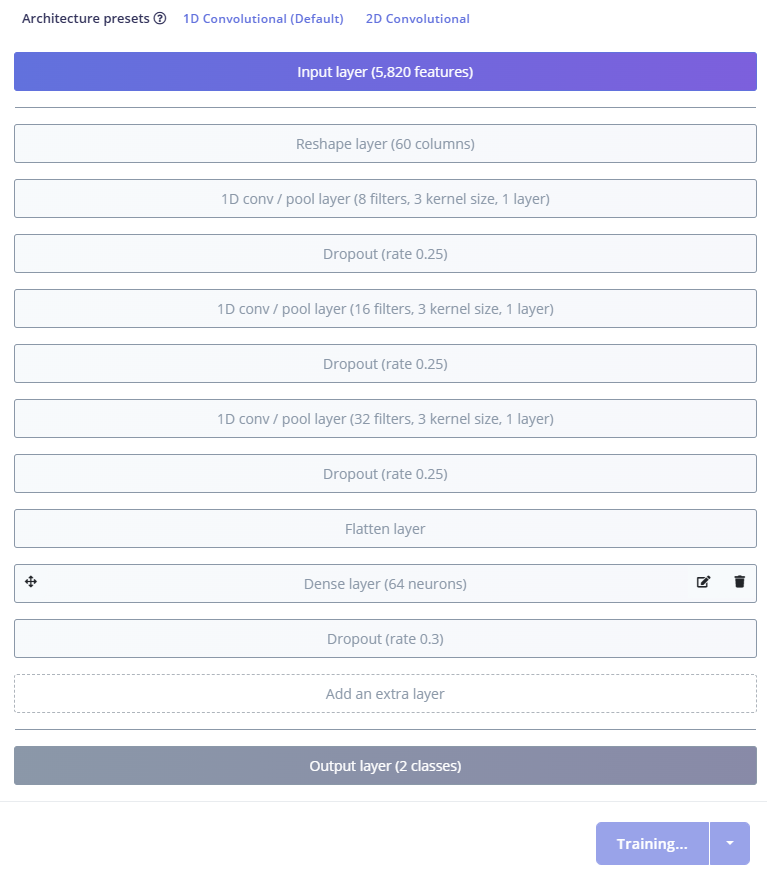
→ Dense Layer（例如 64 units，激活函数 relu）【推荐加】

→ Dropout（如 0.3）

→ Output (2 classes, softmax)

使用的MFE。





Epoch 1/100

91/91 - 9s - loss: 0.4094 - accuracy: 0.8266 - val\_loss: 0.5996 - val\_accuracy: 0.5879 - 9s/epoch - 98ms/step

Epoch 2/100

91/91 - 7s - loss: 0.2180 - accuracy: 0.9128 - val\_loss: 0.4197 - val\_accuracy: 0.6662 - 7s/epoch - 80ms/step

Epoch 3/100

91/91 - 7s - loss: 0.2052 - accuracy: 0.9238 - val\_loss: 0.4653 - val\_accuracy: 0.6387 - 7s/epoch - 81ms/step

Epoch 4/100

Epoch 94% done

91/91 - 7s - loss: 0.1963 - accuracy: 0.9231 - val\_loss: 0.3784 - val\_accuracy: 0.7459 - 7s/epoch - 81ms/step

Epoch 5/100

91/91 - 7s - loss: 0.1877 - accuracy: 0.9210 - val\_loss: 0.3200 - val\_accuracy: 0.7720 - 7s/epoch - 81ms/step

Epoch 6/100

91/91 - 7s - loss: 0.2076 - accuracy: 0.9207 - val\_loss: 0.3635 - val\_accuracy: 0.8846 - 7s/epoch - 82ms/step

Epoch 7/100

91/91 - 7s - loss: 0.2076 - accuracy: 0.9207 - val\_loss: 0.3635 - val\_accuracy: 0.8846 - 7s/epoch - 82ms/step

Epoch 7/100

91/91 - 7s - loss: 0.1929 - accuracy: 0.9306 - val\_loss: 0.3333 - val\_accuracy: 0.9025 - 7s/epoch - 79ms/step

Epoch 8/100

91/91 - 7s - loss: 0.1723 - accuracy: 0.9282 - val\_loss: 0.3088 - val\_accuracy: 0.9560 - 7s/epoch - 81ms/step

Epoch 9/100

91/91 - 7s - loss: 0.1853 - accuracy: 0.9317 - val\_loss: 0.3460 - val\_accuracy: 0.9409 - 7s/epoch - 82ms/step

Epoch 10/100

91/91 - 7s - loss: 0.1861 - accuracy: 0.9303 - val\_loss: 0.3788 - val\_accuracy: 0.8420 - 7s/epoch - 82ms/step

Epoch 11/100

91/91 - 7s - loss: 0.1677 - accuracy: 0.9279 - val\_loss: 0.3027 - val\_accuracy: 0.9093 - 7s/epoch - 80ms/step

Epoch 12/100

91/91 - 7s - loss: 0.1766 - accuracy: 0.9361 - val\_loss: 0.2429 - val\_accuracy: 0.9670 - 7s/epoch - 80ms/step

Epoch 13/100

91/91 - 7s - loss: 0.1708 - accuracy: 0.9368 - val\_loss: 0.4256 - val\_accuracy: 0.9025 - 7s/epoch - 81ms/step

Epoch 14/100

Epoch 95% done

91/91 - 7s - loss: 0.1671 - accuracy: 0.9361 - val\_loss: 0.2465 - val\_accuracy: 0.9588 - 7s/epoch - 80ms/step

Epoch 15/100

91/91 - 7s - loss: 0.2086 - accuracy: 0.9231 - val\_loss: 0.2181 - val\_accuracy: 0.9588 - 7s/epoch - 81ms/step

Epoch 16/100

91/91 - 7s - loss: 0.1974 - accuracy: 0.9227 - val\_loss: 0.3080 - val\_accuracy: 0.9602 - 7s/epoch - 80ms/step

Epoch 17/100

91/91 - 7s - loss: 0.1700 - accuracy: 0.9354 - val\_loss: 0.2714 - val\_accuracy: 0.8887 - 7s/epoch - 79ms/step

Epoch 18/100

91/91 - 7s - loss: 0.1826 - accuracy: 0.9200 - val\_loss: 0.5880 - val\_accuracy: 0.7596 - 7s/epoch - 81ms/step

Epoch 19/100

91/91 - 7s - loss: 0.1792 - accuracy: 0.9293 - val\_loss: 0.3600 - val\_accuracy: 0.8571 - 7s/epoch - 80ms/step

Epoch 20/100

91/91 - 7s - loss: 0.1870 - accuracy: 0.9320 - val\_loss: 0.4814 - val\_accuracy: 0.7005 - 7s/epoch - 80ms/step

Epoch 21/100

91/91 - 7s - loss: 0.1975 - accuracy: 0.9286 - val\_loss: 0.2947 - val\_accuracy: 0.8104 - 7s/epoch - 81ms/step

Epoch 22/100

91/91 - 7s - loss: 0.1878 - accuracy: 0.9265 - val\_loss: 0.2336 - val\_accuracy: 0.9148 - 7s/epoch - 81ms/step

Epoch 23/100

91/91 - 8s - loss: 0.1858 - accuracy: 0.9299 - val\_loss: 0.2173 - val\_accuracy: 0.9519 - 8s/epoch - 82ms/step

Epoch 24/100

91/91 - 8s - loss: 0.1858 - accuracy: 0.9299 - val\_loss: 0.2173 - val\_accuracy: 0.9519 - 8s/epoch - 82ms/step

Epoch 24/100

91/91 - 8s - loss: 0.1798 - accuracy: 0.9313 - val\_loss: 0.2790 - val\_accuracy: 0.9258 - 8s/epoch - 84ms/step

Epoch 25/100

91/91 - 8s - loss: 0.1614 - accuracy: 0.9341 - val\_loss: 0.4944 - val\_accuracy: 0.7115 - 8s/epoch - 83ms/step

Epoch 26/100

91/91 - 8s - loss: 0.1695 - accuracy: 0.9382 - val\_loss: 0.4714 - val\_accuracy: 0.7019 - 8s/epoch - 82ms/step

Epoch 27/100

91/91 - 7s - loss: 0.1675 - accuracy: 0.9365 - val\_loss: 0.2832 - val\_accuracy: 0.9409 - 7s/epoch - 80ms/step

Epoch 28/100

Epoch 89% done

91/91 - 8s - loss: 0.1664 - accuracy: 0.9299 - val\_loss: 0.3960 - val\_accuracy: 0.8654 - 8s/epoch - 83ms/step

Epoch 29/100

91/91 - 7s - loss: 0.1982 - accuracy: 0.9241 - val\_loss: 0.3909 - val\_accuracy: 0.7459 - 7s/epoch - 81ms/step

Epoch 30/100

91/91 - 7s - loss: 0.1758 - accuracy: 0.9275 - val\_loss: 0.5271 - val\_accuracy: 0.8448 - 7s/epoch - 82ms/step

Epoch 31/100

91/91 - 7s - loss: 0.1653 - accuracy: 0.9306 - val\_loss: 0.5326 - val\_accuracy: 0.6868 - 7s/epoch - 79ms/step

Epoch 32/100

91/91 - 7s - loss: 0.1794 - accuracy: 0.9293 - val\_loss: 0.4058 - val\_accuracy: 0.6621 - 7s/epoch - 79ms/step

Epoch 33/100

91/91 - 7s - loss: 0.1794 - accuracy: 0.9293 - val\_loss: 0.4058 - val\_accuracy: 0.6621 - 7s/epoch - 79ms/step

Epoch 33/100

91/91 - 7s - loss: 0.1868 - accuracy: 0.9286 - val\_loss: 0.4602 - val\_accuracy: 0.6030 - 7s/epoch - 80ms/step

Epoch 34/100

91/91 - 7s - loss: 0.1747 - accuracy: 0.9293 - val\_loss: 0.4783 - val\_accuracy: 0.6030 - 7s/epoch - 80ms/step

Epoch 35/100

91/91 - 7s - loss: 0.1783 - accuracy: 0.9251 - val\_loss: 0.4855 - val\_accuracy: 0.6003 - 7s/epoch - 80ms/step

Epoch 36/100

91/91 - 7s - loss: 0.1786 - accuracy: 0.9375 - val\_loss: 0.3296 - val\_accuracy: 0.9066 - 7s/epoch - 79ms/step

Epoch 37/100

91/91 - 7s - loss: 0.1698 - accuracy: 0.9399 - val\_loss: 0.5353 - val\_accuracy: 0.6195 - 7s/epoch - 81ms/step

Epoch 38/100

91/91 - 7s - loss: 0.1746 - accuracy: 0.9365 - val\_loss: 0.4238 - val\_accuracy: 0.6511 - 7s/epoch - 80ms/step

Epoch 39/100

91/91 - 7s - loss: 0.1908 - accuracy: 0.9306 - val\_loss: 0.3981 - val\_accuracy: 0.6401 - 7s/epoch - 80ms/step

Epoch 40/100

91/91 - 7s - loss: 0.1640 - accuracy: 0.9354 - val\_loss: 0.3700 - val\_accuracy: 0.6621 - 7s/epoch - 81ms/step

Epoch 41/100

91/91 - 8s - loss: 0.1714 - accuracy: 0.9354 - val\_loss: 0.3552 - val\_accuracy: 0.8901 - 8s/epoch - 82ms/step

Epoch 42/100

91/91 - 7s - loss: 0.1722 - accuracy: 0.9375 - val\_loss: 0.4809 - val\_accuracy: 0.8681 - 7s/epoch - 81ms/step

Epoch 43/100

91/91 - 7s - loss: 0.1724 - accuracy: 0.9251 - val\_loss: 0.3866 - val\_accuracy: 0.6937 - 7s/epoch - 80ms/step

Epoch 44/100

91/91 - 8s - loss: 0.1743 - accuracy: 0.9330 - val\_loss: 0.3592 - val\_accuracy: 0.7088 - 8s/epoch - 83ms/step

Epoch 45/100

91/91 - 7s - loss: 0.1722 - accuracy: 0.9320 - val\_loss: 0.3447 - val\_accuracy: 0.6758 - 7s/epoch - 80ms/step

Epoch 46/100

91/91 - 7s - loss: 0.1617 - accuracy: 0.9327 - val\_loss: 0.4311 - val\_accuracy: 0.6538 - 7s/epoch - 82ms/step

Epoch 47/100

91/91 - 7s - loss: 0.1792 - accuracy: 0.9358 - val\_loss: 0.5110 - val\_accuracy: 0.6841 - 7s/epoch - 81ms/step

Epoch 48/100

91/91 - 7s - loss: 0.1636 - accuracy: 0.9382 - val\_loss: 0.4670 - val\_accuracy: 0.6593 - 7s/epoch - 81ms/step

Epoch 49/100

91/91 - 7s - loss: 0.1780 - accuracy: 0.9337 - val\_loss: 0.3485 - val\_accuracy: 0.7390 - 7s/epoch - 81ms/step

Epoch 50/100

91/91 - 8s - loss: 0.1761 - accuracy: 0.9348 - val\_loss: 0.3453 - val\_accuracy: 0.7390 - 8s/epoch - 83ms/step

Epoch 51/100

91/91 - 7s - loss: 0.1654 - accuracy: 0.9341 - val\_loss: 0.3120 - val\_accuracy: 0.8448 - 7s/epoch - 81ms/step

Epoch 52/100

91/91 - 7s - loss: 0.1688 - accuracy: 0.9354 - val\_loss: 0.2225 - val\_accuracy: 0.9684 - 7s/epoch - 81ms/step

Epoch 53/100

91/91 - 7s - loss: 0.1804 - accuracy: 0.9396 - val\_loss: 0.3547 - val\_accuracy: 0.9492 - 7s/epoch - 80ms/step

Epoch 54/100

91/91 - 7s - loss: 0.1699 - accuracy: 0.9323 - val\_loss: 0.2129 - val\_accuracy: 0.9602 - 7s/epoch - 81ms/step

Epoch 55/100

91/91 - 7s - loss: 0.1600 - accuracy: 0.9358 - val\_loss: 0.2089 - val\_accuracy: 0.9753 - 7s/epoch - 81ms/step

Epoch 56/100

91/91 - 7s - loss: 0.1835 - accuracy: 0.9406 - val\_loss: 0.2760 - val\_accuracy: 0.7885 - 7s/epoch - 80ms/step

Epoch 57/100

91/91 - 7s - loss: 0.1629 - accuracy: 0.9341 - val\_loss: 0.2800 - val\_accuracy: 0.7871 - 7s/epoch - 81ms/step

Epoch 58/100

91/91 - 7s - loss: 0.1678 - accuracy: 0.9348 - val\_loss: 0.2843 - val\_accuracy: 0.9203 - 7s/epoch - 81ms/step

Epoch 59/100

Epoch 58/100

91/91 - 7s - loss: 0.1678 - accuracy: 0.9348 - val\_loss: 0.2843 - val\_accuracy: 0.9203 - 7s/epoch - 81ms/step

91/91 - 7s - loss: 0.1655 - accuracy: 0.9348 - val\_loss: 0.3008 - val\_accuracy: 0.9396 - 7s/epoch - 80ms/step

Epoch 60/100

91/91 - 7s - loss: 0.1665 - accuracy: 0.9365 - val\_loss: 0.2694 - val\_accuracy: 0.8448 - 7s/epoch - 80ms/step

Epoch 61/100

91/91 - 7s - loss: 0.1909 - accuracy: 0.9303 - val\_loss: 0.3343 - val\_accuracy: 0.7898 - 7s/epoch - 79ms/step

Epoch 62/100

91/91 - 7s - loss: 0.1696 - accuracy: 0.9365 - val\_loss: 0.2512 - val\_accuracy: 0.8146 - 7s/epoch - 79ms/step

Epoch 63/100

91/91 - 7s - loss: 0.1854 - accuracy: 0.9303 - val\_loss: 0.2658 - val\_accuracy: 0.8297 - 7s/epoch - 80ms/step

Epoch 64/100

91/91 - 7s - loss: 0.1539 - accuracy: 0.9402 - val\_loss: 0.2222 - val\_accuracy: 0.8984 - 7s/epoch - 80ms/step

Epoch 65/100

91/91 - 7s - loss: 0.1827 - accuracy: 0.9293 - val\_loss: 0.2625 - val\_accuracy: 0.9464 - 7s/epoch - 79ms/step

Epoch 66/100

91/91 - 7s - loss: 0.1752 - accuracy: 0.9354 - val\_loss: 0.2448 - val\_accuracy: 0.9245 - 7s/epoch - 80ms/step

Epoch 67/100

Epoch 92% done

91/91 - 7s - loss: 0.1738 - accuracy: 0.9344 - val\_loss: 0.3043 - val\_accuracy: 0.9643 - 7s/epoch - 80ms/step

Epoch 68/100

Epoch 92% done

91/91 - 7s - loss: 0.1738 - accuracy: 0.9344 - val\_loss: 0.3043 - val\_accuracy: 0.9643 - 7s/epoch - 80ms/step

91/91 - 7s - loss: 0.2157 - accuracy: 0.9159 - val\_loss: 0.1824 - val\_accuracy: 0.9753 - 7s/epoch - 80ms/step

Epoch 69/100

Epoch 91% done

91/91 - 7s - loss: 0.1783 - accuracy: 0.9269 - val\_loss: 0.2721 - val\_accuracy: 0.8709 - 7s/epoch - 80ms/step

Epoch 70/100

Epoch 93% done

91/91 - 7s - loss: 0.1695 - accuracy: 0.9296 - val\_loss: 0.2069 - val\_accuracy: 0.9725 - 7s/epoch - 80ms/step

Epoch 71/100

91/91 - 7s - loss: 0.1809 - accuracy: 0.9293 - val\_loss: 0.2716 - val\_accuracy: 0.9533 - 7s/epoch - 79ms/step

Epoch 72/100

91/91 - 7s - loss: 0.1718 - accuracy: 0.9217 - val\_loss: 0.2662 - val\_accuracy: 0.9492 - 7s/epoch - 81ms/step

Epoch 73/100

91/91 - 7s - loss: 0.1774 - accuracy: 0.9323 - val\_loss: 0.4017 - val\_accuracy: 0.8297 - 7s/epoch - 78ms/step

Epoch 74/100

91/91 - 7s - loss: 0.1759 - accuracy: 0.9323 - val\_loss: 0.2408 - val\_accuracy: 0.9313 - 7s/epoch - 81ms/step

Epoch 75/100

91/91 - 7s - loss: 0.1696 - accuracy: 0.9337 - val\_loss: 0.2716 - val\_accuracy: 0.9354 - 7s/epoch - 82ms/step

Epoch 76/100

91/91 - 7s - loss: 0.1540 - accuracy: 0.9334 - val\_loss: 0.2583 - val\_accuracy: 0.9354 - 7s/epoch - 79ms/step

Epoch 77/100

91/91 - 7s - loss: 0.1619 - accuracy: 0.9354 - val\_loss: 0.2348 - val\_accuracy: 0.9217 - 7s/epoch - 80ms/step

Epoch 78/100

91/91 - 7s - loss: 0.1687 - accuracy: 0.9341 - val\_loss: 0.2226 - val\_accuracy: 0.9492 - 7s/epoch - 82ms/step

Epoch 79/100

91/91 - 7s - loss: 0.1652 - accuracy: 0.9330 - val\_loss: 0.2123 - val\_accuracy: 0.9299 - 7s/epoch - 79ms/step

Epoch 80/100

91/91 - 7s - loss: 0.1733 - accuracy: 0.9334 - val\_loss: 0.2372 - val\_accuracy: 0.9396 - 7s/epoch - 79ms/step

Epoch 81/100

91/91 - 7s - loss: 0.1844 - accuracy: 0.9303 - val\_loss: 0.2794 - val\_accuracy: 0.9245 - 7s/epoch - 78ms/step

Epoch 82/100

91/91 - 7s - loss: 0.1711 - accuracy: 0.9289 - val\_loss: 0.2871 - val\_accuracy: 0.9327 - 7s/epoch - 78ms/step

Epoch 83/100

91/91 - 7s - loss: 0.1561 - accuracy: 0.9378 - val\_loss: 0.2440 - val\_accuracy: 0.9505 - 7s/epoch - 78ms/step

Epoch 84/100

91/91 - 7s - loss: 0.1929 - accuracy: 0.9365 - val\_loss: 0.2081 - val\_accuracy: 0.9382 - 7s/epoch - 77ms/step

Epoch 85/100

91/91 - 7s - loss: 0.1550 - accuracy: 0.9368 - val\_loss: 0.2321 - val\_accuracy: 0.9574 - 7s/epoch - 79ms/step

Epoch 86/100

91/91 - 7s - loss: 0.1548 - accuracy: 0.9392 - val\_loss: 0.1999 - val\_accuracy: 0.9492 - 7s/epoch - 80ms/step

Epoch 87/100

91/91 - 7s - loss: 0.1549 - accuracy: 0.9382 - val\_loss: 0.2007 - val\_accuracy: 0.9560 - 7s/epoch - 80ms/step

Epoch 88/100

91/91 - 7s - loss: 0.1892 - accuracy: 0.9248 - val\_loss: 0.2844 - val\_accuracy: 0.9464 - 7s/epoch - 80ms/step

Epoch 89/100

91/91 - 7s - loss: 0.1660 - accuracy: 0.9358 - val\_loss: 0.2346 - val\_accuracy: 0.9437 - 7s/epoch - 81ms/step

Epoch 90/100

91/91 - 7s - loss: 0.1615 - accuracy: 0.9299 - val\_loss: 0.2462 - val\_accuracy: 0.9505 - 7s/epoch - 79ms/step

Epoch 91/100

91/91 - 7s - loss: 0.1645 - accuracy: 0.9365 - val\_loss: 0.3747 - val\_accuracy: 0.8997 - 7s/epoch - 81ms/step

Epoch 92/100

91/91 - 7s - loss: 0.1744 - accuracy: 0.9351 - val\_loss: 0.4291 - val\_accuracy: 0.9533 - 7s/epoch - 80ms/step

Epoch 93/100

91/91 - 7s - loss: 0.1525 - accuracy: 0.9416 - val\_loss: 0.3768 - val\_accuracy: 0.9574 - 7s/epoch - 80ms/step

Epoch 94/100

91/91 - 7s - loss: 0.1572 - accuracy: 0.9392 - val\_loss: 0.2867 - val\_accuracy: 0.9615 - 7s/epoch - 78ms/step

Epoch 95/100

91/91 - 7s - loss: 0.1571 - accuracy: 0.9444 - val\_loss: 0.3011 - val\_accuracy: 0.8956 - 7s/epoch - 78ms/step

Epoch 96/100

91/91 - 7s - loss: 0.1765 - accuracy: 0.9337 - val\_loss: 0.2230 - val\_accuracy: 0.9533 - 7s/epoch - 78ms/step

Epoch 97/100

91/91 - 7s - loss: 0.1728 - accuracy: 0.9337 - val\_loss: 0.1771 - val\_accuracy: 0.9464 - 7s/epoch - 78ms/step

Epoch 98/100

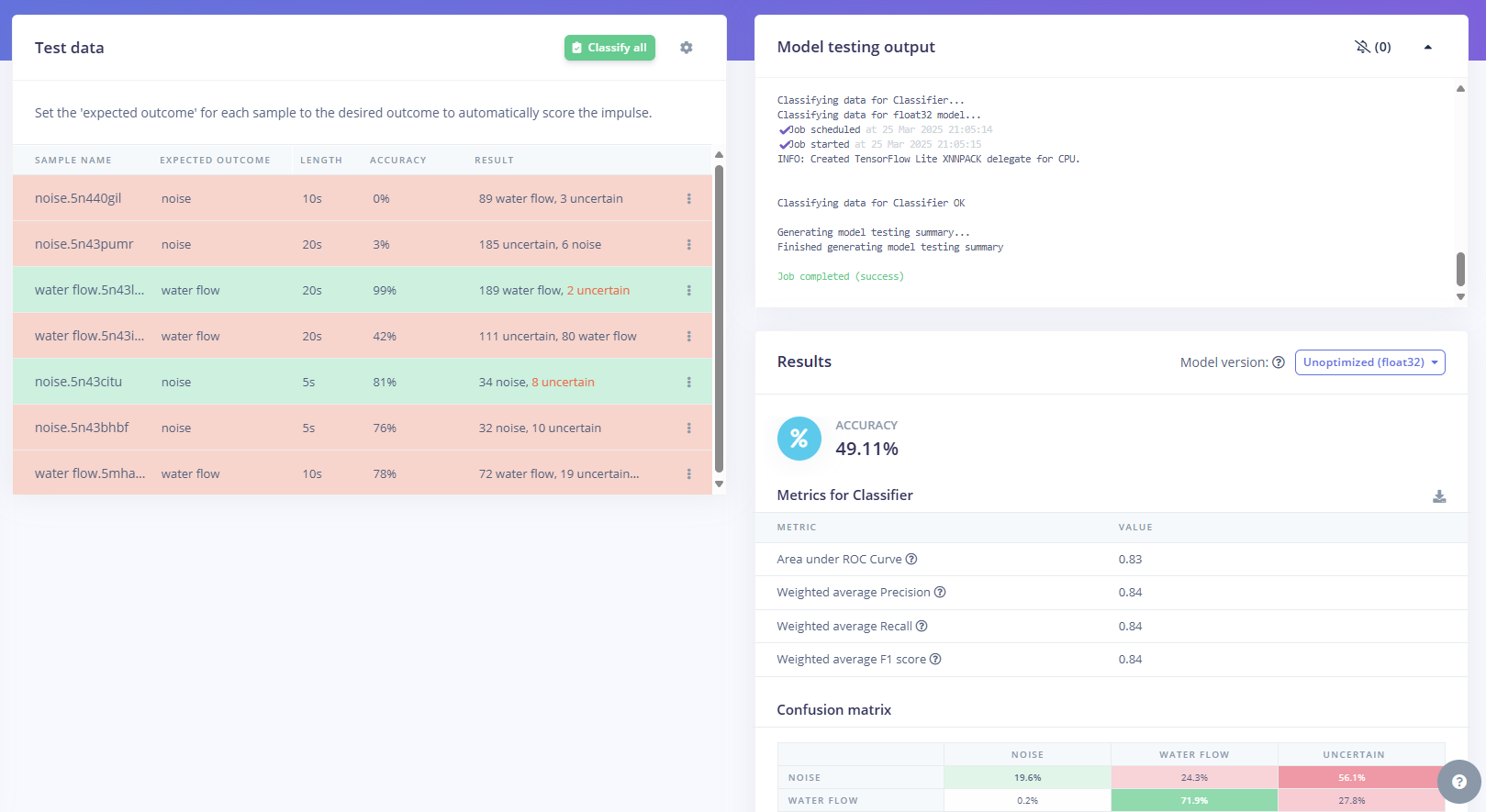
91/91 - 7s - loss: 0.1821 - accuracy: 0.9279 - val\_loss: 0.2120 - val\_accuracy: 0.9464 - 7s/epoch - 78ms/step

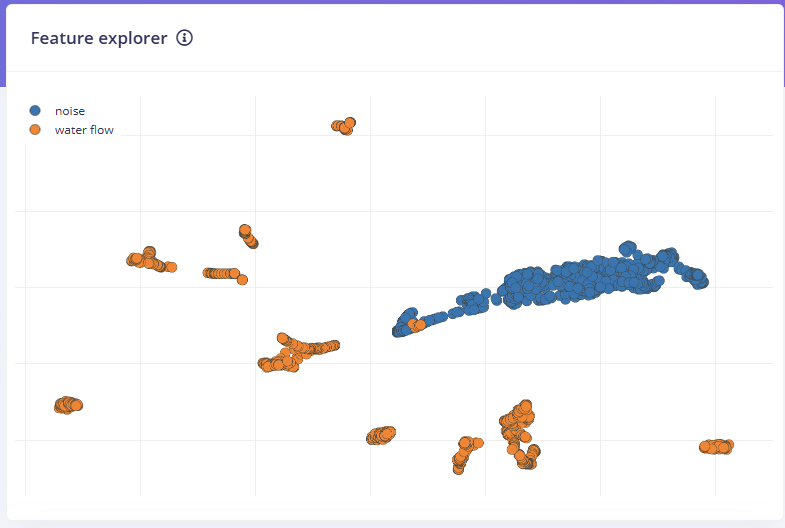
Epoch 99/100

91/91 - 7s - loss: 0.1600 - accuracy: 0.9330 - val\_loss: 0.1858 - val\_accuracy: 0.9451 - 7s/epoch - 78ms/step

Epoch 100/100

91/91 - 7s - loss: 0.1619 - accuracy: 0.9382 - val\_loss: 0.2028 - val\_accuracy: 0.9519 - 7s/epoch - 80ms/step

Finished training  
我需要先做一样的事情，用100epoch找出最佳epoch是多少。同时也可以将这个模型先保存，测试其正确率。

好吧这个模型完全没有保留的必要，算了吧。错误比较多的是捏瓶子的噪音和有很强底噪的安静房间。我认为是底噪的问题，所以我把low frequency改为300并把Noise floor (dB)改为-60。  
感觉还行。将epoch改为50进行尝试

Epoch 1/100

91/91 - 9s - loss: 0.4094 - accuracy: 0.8266 - val\_loss: 0.5996 - val\_accuracy: 0.5879 - 9s/epoch - 98ms/step

Epoch 2/100

91/91 - 7s - loss: 0.2180 - accuracy: 0.9128 - val\_loss: 0.4197 - val\_accuracy: 0.6662 - 7s/epoch - 80ms/step

Epoch 3/100

91/91 - 7s - loss: 0.2052 - accuracy: 0.9238 - val\_loss: 0.4653 - val\_accuracy: 0.6387 - 7s/epoch - 81ms/step

Epoch 4/100

Epoch 94% done

91/91 - 7s - loss: 0.1963 - accuracy: 0.9231 - val\_loss: 0.3784 - val\_accuracy: 0.7459 - 7s/epoch - 81ms/step

Epoch 5/100

91/91 - 7s - loss: 0.1877 - accuracy: 0.9210 - val\_loss: 0.3200 - val\_accuracy: 0.7720 - 7s/epoch - 81ms/step

Epoch 6/100

91/91 - 7s - loss: 0.2076 - accuracy: 0.9207 - val\_loss: 0.3635 - val\_accuracy: 0.8846 - 7s/epoch - 82ms/step

Epoch 7/100

91/91 - 7s - loss: 0.2076 - accuracy: 0.9207 - val\_loss: 0.3635 - val\_accuracy: 0.8846 - 7s/epoch - 82ms/step

Epoch 7/100

91/91 - 7s - loss: 0.1929 - accuracy: 0.9306 - val\_loss: 0.3333 - val\_accuracy: 0.9025 - 7s/epoch - 79ms/step

Epoch 8/100

91/91 - 7s - loss: 0.1723 - accuracy: 0.9282 - val\_loss: 0.3088 - val\_accuracy: 0.9560 - 7s/epoch - 81ms/step

Epoch 9/100

91/91 - 7s - loss: 0.1853 - accuracy: 0.9317 - val\_loss: 0.3460 - val\_accuracy: 0.9409 - 7s/epoch - 82ms/step

Epoch 10/100

91/91 - 7s - loss: 0.1861 - accuracy: 0.9303 - val\_loss: 0.3788 - val\_accuracy: 0.8420 - 7s/epoch - 82ms/step

Epoch 11/100

91/91 - 7s - loss: 0.1677 - accuracy: 0.9279 - val\_loss: 0.3027 - val\_accuracy: 0.9093 - 7s/epoch - 80ms/step

Epoch 12/100

91/91 - 7s - loss: 0.1766 - accuracy: 0.9361 - val\_loss: 0.2429 - val\_accuracy: 0.9670 - 7s/epoch - 80ms/step

Epoch 13/100

91/91 - 7s - loss: 0.1708 - accuracy: 0.9368 - val\_loss: 0.4256 - val\_accuracy: 0.9025 - 7s/epoch - 81ms/step

Epoch 14/100

Epoch 95% done

91/91 - 7s - loss: 0.1671 - accuracy: 0.9361 - val\_loss: 0.2465 - val\_accuracy: 0.9588 - 7s/epoch - 80ms/step

Epoch 15/100

91/91 - 7s - loss: 0.2086 - accuracy: 0.9231 - val\_loss: 0.2181 - val\_accuracy: 0.9588 - 7s/epoch - 81ms/step

Epoch 16/100

91/91 - 7s - loss: 0.1974 - accuracy: 0.9227 - val\_loss: 0.3080 - val\_accuracy: 0.9602 - 7s/epoch - 80ms/step

Epoch 17/100

91/91 - 7s - loss: 0.1700 - accuracy: 0.9354 - val\_loss: 0.2714 - val\_accuracy: 0.8887 - 7s/epoch - 79ms/step

Epoch 18/100

91/91 - 7s - loss: 0.1826 - accuracy: 0.9200 - val\_loss: 0.5880 - val\_accuracy: 0.7596 - 7s/epoch - 81ms/step

Epoch 19/100

91/91 - 7s - loss: 0.1792 - accuracy: 0.9293 - val\_loss: 0.3600 - val\_accuracy: 0.8571 - 7s/epoch - 80ms/step

Epoch 20/100

91/91 - 7s - loss: 0.1870 - accuracy: 0.9320 - val\_loss: 0.4814 - val\_accuracy: 0.7005 - 7s/epoch - 80ms/step

Epoch 21/100

91/91 - 7s - loss: 0.1975 - accuracy: 0.9286 - val\_loss: 0.2947 - val\_accuracy: 0.8104 - 7s/epoch - 81ms/step

Epoch 22/100

91/91 - 7s - loss: 0.1878 - accuracy: 0.9265 - val\_loss: 0.2336 - val\_accuracy: 0.9148 - 7s/epoch - 81ms/step

Epoch 23/100

91/91 - 8s - loss: 0.1858 - accuracy: 0.9299 - val\_loss: 0.2173 - val\_accuracy: 0.9519 - 8s/epoch - 82ms/step

Epoch 24/100

91/91 - 8s - loss: 0.1858 - accuracy: 0.9299 - val\_loss: 0.2173 - val\_accuracy: 0.9519 - 8s/epoch - 82ms/step

Epoch 24/100

91/91 - 8s - loss: 0.1798 - accuracy: 0.9313 - val\_loss: 0.2790 - val\_accuracy: 0.9258 - 8s/epoch - 84ms/step

Epoch 25/100

91/91 - 8s - loss: 0.1614 - accuracy: 0.9341 - val\_loss: 0.4944 - val\_accuracy: 0.7115 - 8s/epoch - 83ms/step

Epoch 26/100

91/91 - 8s - loss: 0.1695 - accuracy: 0.9382 - val\_loss: 0.4714 - val\_accuracy: 0.7019 - 8s/epoch - 82ms/step

Epoch 27/100

91/91 - 7s - loss: 0.1675 - accuracy: 0.9365 - val\_loss: 0.2832 - val\_accuracy: 0.9409 - 7s/epoch - 80ms/step

Epoch 28/100

Epoch 89% done

91/91 - 8s - loss: 0.1664 - accuracy: 0.9299 - val\_loss: 0.3960 - val\_accuracy: 0.8654 - 8s/epoch - 83ms/step

Epoch 29/100

91/91 - 7s - loss: 0.1982 - accuracy: 0.9241 - val\_loss: 0.3909 - val\_accuracy: 0.7459 - 7s/epoch - 81ms/step

Epoch 30/100

91/91 - 7s - loss: 0.1758 - accuracy: 0.9275 - val\_loss: 0.5271 - val\_accuracy: 0.8448 - 7s/epoch - 82ms/step

Epoch 31/100

91/91 - 7s - loss: 0.1653 - accuracy: 0.9306 - val\_loss: 0.5326 - val\_accuracy: 0.6868 - 7s/epoch - 79ms/step

Epoch 32/100

91/91 - 7s - loss: 0.1794 - accuracy: 0.9293 - val\_loss: 0.4058 - val\_accuracy: 0.6621 - 7s/epoch - 79ms/step

Epoch 33/100

91/91 - 7s - loss: 0.1794 - accuracy: 0.9293 - val\_loss: 0.4058 - val\_accuracy: 0.6621 - 7s/epoch - 79ms/step

Epoch 33/100

91/91 - 7s - loss: 0.1868 - accuracy: 0.9286 - val\_loss: 0.4602 - val\_accuracy: 0.6030 - 7s/epoch - 80ms/step

Epoch 34/100

91/91 - 7s - loss: 0.1747 - accuracy: 0.9293 - val\_loss: 0.4783 - val\_accuracy: 0.6030 - 7s/epoch - 80ms/step

Epoch 35/100

91/91 - 7s - loss: 0.1783 - accuracy: 0.9251 - val\_loss: 0.4855 - val\_accuracy: 0.6003 - 7s/epoch - 80ms/step

Epoch 36/100

91/91 - 7s - loss: 0.1786 - accuracy: 0.9375 - val\_loss: 0.3296 - val\_accuracy: 0.9066 - 7s/epoch - 79ms/step

Epoch 37/100

91/91 - 7s - loss: 0.1698 - accuracy: 0.9399 - val\_loss: 0.5353 - val\_accuracy: 0.6195 - 7s/epoch - 81ms/step

Epoch 38/100

91/91 - 7s - loss: 0.1746 - accuracy: 0.9365 - val\_loss: 0.4238 - val\_accuracy: 0.6511 - 7s/epoch - 80ms/step

Epoch 39/100

91/91 - 7s - loss: 0.1908 - accuracy: 0.9306 - val\_loss: 0.3981 - val\_accuracy: 0.6401 - 7s/epoch - 80ms/step

Epoch 40/100

91/91 - 7s - loss: 0.1640 - accuracy: 0.9354 - val\_loss: 0.3700 - val\_accuracy: 0.6621 - 7s/epoch - 81ms/step

Epoch 41/100

91/91 - 8s - loss: 0.1714 - accuracy: 0.9354 - val\_loss: 0.3552 - val\_accuracy: 0.8901 - 8s/epoch - 82ms/step

Epoch 42/100

91/91 - 7s - loss: 0.1722 - accuracy: 0.9375 - val\_loss: 0.4809 - val\_accuracy: 0.8681 - 7s/epoch - 81ms/step

Epoch 43/100

91/91 - 7s - loss: 0.1724 - accuracy: 0.9251 - val\_loss: 0.3866 - val\_accuracy: 0.6937 - 7s/epoch - 80ms/step

Epoch 44/100

91/91 - 8s - loss: 0.1743 - accuracy: 0.9330 - val\_loss: 0.3592 - val\_accuracy: 0.7088 - 8s/epoch - 83ms/step

Epoch 45/100

91/91 - 7s - loss: 0.1722 - accuracy: 0.9320 - val\_loss: 0.3447 - val\_accuracy: 0.6758 - 7s/epoch - 80ms/step

Epoch 46/100

91/91 - 7s - loss: 0.1617 - accuracy: 0.9327 - val\_loss: 0.4311 - val\_accuracy: 0.6538 - 7s/epoch - 82ms/step

Epoch 47/100

91/91 - 7s - loss: 0.1792 - accuracy: 0.9358 - val\_loss: 0.5110 - val\_accuracy: 0.6841 - 7s/epoch - 81ms/step

Epoch 48/100

91/91 - 7s - loss: 0.1636 - accuracy: 0.9382 - val\_loss: 0.4670 - val\_accuracy: 0.6593 - 7s/epoch - 81ms/step

Epoch 49/100

91/91 - 7s - loss: 0.1780 - accuracy: 0.9337 - val\_loss: 0.3485 - val\_accuracy: 0.7390 - 7s/epoch - 81ms/step

Epoch 50/100

91/91 - 8s - loss: 0.1761 - accuracy: 0.9348 - val\_loss: 0.3453 - val\_accuracy: 0.7390 - 8s/epoch - 83ms/step

Epoch 51/100

91/91 - 7s - loss: 0.1654 - accuracy: 0.9341 - val\_loss: 0.3120 - val\_accuracy: 0.8448 - 7s/epoch - 81ms/step

Epoch 52/100

91/91 - 7s - loss: 0.1688 - accuracy: 0.9354 - val\_loss: 0.2225 - val\_accuracy: 0.9684 - 7s/epoch - 81ms/step

Epoch 53/100

91/91 - 7s - loss: 0.1804 - accuracy: 0.9396 - val\_loss: 0.3547 - val\_accuracy: 0.9492 - 7s/epoch - 80ms/step

Epoch 54/100

91/91 - 7s - loss: 0.1699 - accuracy: 0.9323 - val\_loss: 0.2129 - val\_accuracy: 0.9602 - 7s/epoch - 81ms/step

Epoch 55/100

91/91 - 7s - loss: 0.1600 - accuracy: 0.9358 - val\_loss: 0.2089 - val\_accuracy: 0.9753 - 7s/epoch - 81ms/step

Epoch 56/100

91/91 - 7s - loss: 0.1835 - accuracy: 0.9406 - val\_loss: 0.2760 - val\_accuracy: 0.7885 - 7s/epoch - 80ms/step

Epoch 57/100

91/91 - 7s - loss: 0.1629 - accuracy: 0.9341 - val\_loss: 0.2800 - val\_accuracy: 0.7871 - 7s/epoch - 81ms/step

Epoch 58/100

91/91 - 7s - loss: 0.1678 - accuracy: 0.9348 - val\_loss: 0.2843 - val\_accuracy: 0.9203 - 7s/epoch - 81ms/step

Epoch 59/100

Epoch 58/100

91/91 - 7s - loss: 0.1678 - accuracy: 0.9348 - val\_loss: 0.2843 - val\_accuracy: 0.9203 - 7s/epoch - 81ms/step

91/91 - 7s - loss: 0.1655 - accuracy: 0.9348 - val\_loss: 0.3008 - val\_accuracy: 0.9396 - 7s/epoch - 80ms/step

Epoch 60/100

91/91 - 7s - loss: 0.1665 - accuracy: 0.9365 - val\_loss: 0.2694 - val\_accuracy: 0.8448 - 7s/epoch - 80ms/step

Epoch 61/100

91/91 - 7s - loss: 0.1909 - accuracy: 0.9303 - val\_loss: 0.3343 - val\_accuracy: 0.7898 - 7s/epoch - 79ms/step

Epoch 62/100

91/91 - 7s - loss: 0.1696 - accuracy: 0.9365 - val\_loss: 0.2512 - val\_accuracy: 0.8146 - 7s/epoch - 79ms/step

Epoch 63/100

91/91 - 7s - loss: 0.1854 - accuracy: 0.9303 - val\_loss: 0.2658 - val\_accuracy: 0.8297 - 7s/epoch - 80ms/step

Epoch 64/100

91/91 - 7s - loss: 0.1539 - accuracy: 0.9402 - val\_loss: 0.2222 - val\_accuracy: 0.8984 - 7s/epoch - 80ms/step

Epoch 65/100

91/91 - 7s - loss: 0.1827 - accuracy: 0.9293 - val\_loss: 0.2625 - val\_accuracy: 0.9464 - 7s/epoch - 79ms/step

Epoch 66/100

91/91 - 7s - loss: 0.1752 - accuracy: 0.9354 - val\_loss: 0.2448 - val\_accuracy: 0.9245 - 7s/epoch - 80ms/step

Epoch 67/100

Epoch 92% done

91/91 - 7s - loss: 0.1738 - accuracy: 0.9344 - val\_loss: 0.3043 - val\_accuracy: 0.9643 - 7s/epoch - 80ms/step

Epoch 68/100

Epoch 92% done

91/91 - 7s - loss: 0.1738 - accuracy: 0.9344 - val\_loss: 0.3043 - val\_accuracy: 0.9643 - 7s/epoch - 80ms/step

91/91 - 7s - loss: 0.2157 - accuracy: 0.9159 - val\_loss: 0.1824 - val\_accuracy: 0.9753 - 7s/epoch - 80ms/step

Epoch 69/100

Epoch 91% done

91/91 - 7s - loss: 0.1783 - accuracy: 0.9269 - val\_loss: 0.2721 - val\_accuracy: 0.8709 - 7s/epoch - 80ms/step

Epoch 70/100

Epoch 93% done

91/91 - 7s - loss: 0.1695 - accuracy: 0.9296 - val\_loss: 0.2069 - val\_accuracy: 0.9725 - 7s/epoch - 80ms/step

Epoch 71/100

91/91 - 7s - loss: 0.1809 - accuracy: 0.9293 - val\_loss: 0.2716 - val\_accuracy: 0.9533 - 7s/epoch - 79ms/step

Epoch 72/100

91/91 - 7s - loss: 0.1718 - accuracy: 0.9217 - val\_loss: 0.2662 - val\_accuracy: 0.9492 - 7s/epoch - 81ms/step

Epoch 73/100

91/91 - 7s - loss: 0.1774 - accuracy: 0.9323 - val\_loss: 0.4017 - val\_accuracy: 0.8297 - 7s/epoch - 78ms/step

Epoch 74/100

91/91 - 7s - loss: 0.1759 - accuracy: 0.9323 - val\_loss: 0.2408 - val\_accuracy: 0.9313 - 7s/epoch - 81ms/step

Epoch 75/100

91/91 - 7s - loss: 0.1696 - accuracy: 0.9337 - val\_loss: 0.2716 - val\_accuracy: 0.9354 - 7s/epoch - 82ms/step

Epoch 76/100

91/91 - 7s - loss: 0.1540 - accuracy: 0.9334 - val\_loss: 0.2583 - val\_accuracy: 0.9354 - 7s/epoch - 79ms/step

Epoch 77/100

91/91 - 7s - loss: 0.1619 - accuracy: 0.9354 - val\_loss: 0.2348 - val\_accuracy: 0.9217 - 7s/epoch - 80ms/step

Epoch 78/100

91/91 - 7s - loss: 0.1687 - accuracy: 0.9341 - val\_loss: 0.2226 - val\_accuracy: 0.9492 - 7s/epoch - 82ms/step

Epoch 79/100

91/91 - 7s - loss: 0.1652 - accuracy: 0.9330 - val\_loss: 0.2123 - val\_accuracy: 0.9299 - 7s/epoch - 79ms/step

Epoch 80/100

91/91 - 7s - loss: 0.1733 - accuracy: 0.9334 - val\_loss: 0.2372 - val\_accuracy: 0.9396 - 7s/epoch - 79ms/step

Epoch 81/100

91/91 - 7s - loss: 0.1844 - accuracy: 0.9303 - val\_loss: 0.2794 - val\_accuracy: 0.9245 - 7s/epoch - 78ms/step

Epoch 82/100

91/91 - 7s - loss: 0.1711 - accuracy: 0.9289 - val\_loss: 0.2871 - val\_accuracy: 0.9327 - 7s/epoch - 78ms/step

Epoch 83/100

91/91 - 7s - loss: 0.1561 - accuracy: 0.9378 - val\_loss: 0.2440 - val\_accuracy: 0.9505 - 7s/epoch - 78ms/step

Epoch 84/100

91/91 - 7s - loss: 0.1929 - accuracy: 0.9365 - val\_loss: 0.2081 - val\_accuracy: 0.9382 - 7s/epoch - 77ms/step

Epoch 85/100

91/91 - 7s - loss: 0.1550 - accuracy: 0.9368 - val\_loss: 0.2321 - val\_accuracy: 0.9574 - 7s/epoch - 79ms/step

Epoch 86/100

91/91 - 7s - loss: 0.1548 - accuracy: 0.9392 - val\_loss: 0.1999 - val\_accuracy: 0.9492 - 7s/epoch - 80ms/step

Epoch 87/100

91/91 - 7s - loss: 0.1549 - accuracy: 0.9382 - val\_loss: 0.2007 - val\_accuracy: 0.9560 - 7s/epoch - 80ms/step

Epoch 88/100

91/91 - 7s - loss: 0.1892 - accuracy: 0.9248 - val\_loss: 0.2844 - val\_accuracy: 0.9464 - 7s/epoch - 80ms/step

Epoch 89/100

91/91 - 7s - loss: 0.1660 - accuracy: 0.9358 - val\_loss: 0.2346 - val\_accuracy: 0.9437 - 7s/epoch - 81ms/step

Epoch 90/100

91/91 - 7s - loss: 0.1615 - accuracy: 0.9299 - val\_loss: 0.2462 - val\_accuracy: 0.9505 - 7s/epoch - 79ms/step

Epoch 91/100

91/91 - 7s - loss: 0.1645 - accuracy: 0.9365 - val\_loss: 0.3747 - val\_accuracy: 0.8997 - 7s/epoch - 81ms/step

Epoch 92/100

91/91 - 7s - loss: 0.1744 - accuracy: 0.9351 - val\_loss: 0.4291 - val\_accuracy: 0.9533 - 7s/epoch - 80ms/step

Epoch 93/100

91/91 - 7s - loss: 0.1525 - accuracy: 0.9416 - val\_loss: 0.3768 - val\_accuracy: 0.9574 - 7s/epoch - 80ms/step

Epoch 94/100

91/91 - 7s - loss: 0.1572 - accuracy: 0.9392 - val\_loss: 0.2867 - val\_accuracy: 0.9615 - 7s/epoch - 78ms/step

Epoch 95/100

91/91 - 7s - loss: 0.1571 - accuracy: 0.9444 - val\_loss: 0.3011 - val\_accuracy: 0.8956 - 7s/epoch - 78ms/step

Epoch 96/100

91/91 - 7s - loss: 0.1765 - accuracy: 0.9337 - val\_loss: 0.2230 - val\_accuracy: 0.9533 - 7s/epoch - 78ms/step

Epoch 97/100

91/91 - 7s - loss: 0.1728 - accuracy: 0.9337 - val\_loss: 0.1771 - val\_accuracy: 0.9464 - 7s/epoch - 78ms/step

Epoch 98/100

91/91 - 7s - loss: 0.1821 - accuracy: 0.9279 - val\_loss: 0.2120 - val\_accuracy: 0.9464 - 7s/epoch - 78ms/step

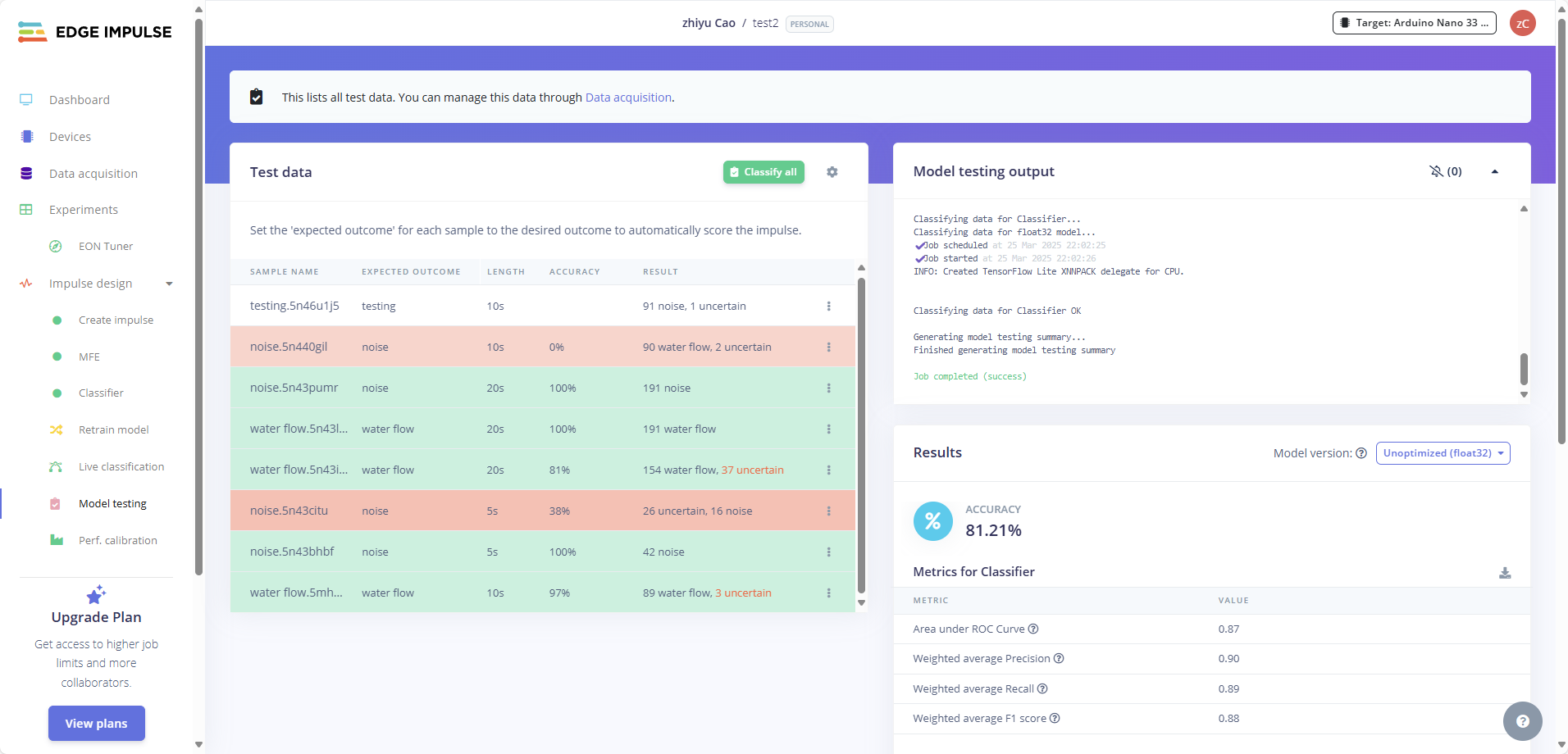
Epoch 99/100

91/91 - 7s - loss: 0.1600 - accuracy: 0.9330 - val\_loss: 0.1858 - val\_accuracy: 0.9451 - 7s/epoch - 78ms/step

Epoch 100/100

91/91 - 7s - loss: 0.1619 - accuracy: 0.9382 - val\_loss: 0.2028 - val\_accuracy: 0.9519 - 7s/epoch - 80ms/step

懒得分析了，晚点再说

测试正确率显著提升！ 我是天才！但能看到，底噪过高的问题还是没被解决。  
  
 训练完成后，在 **"Model testing" 页面** 或 **"NN Classifier" 页面**底部：

 点击 **“View training performance”**  
→ 可以看到每一轮 epoch 的 loss 和 accuracy

 Edge Impulse 会**自动保存表现最好的模型（通常 val\_accuracy 最高的那轮）**

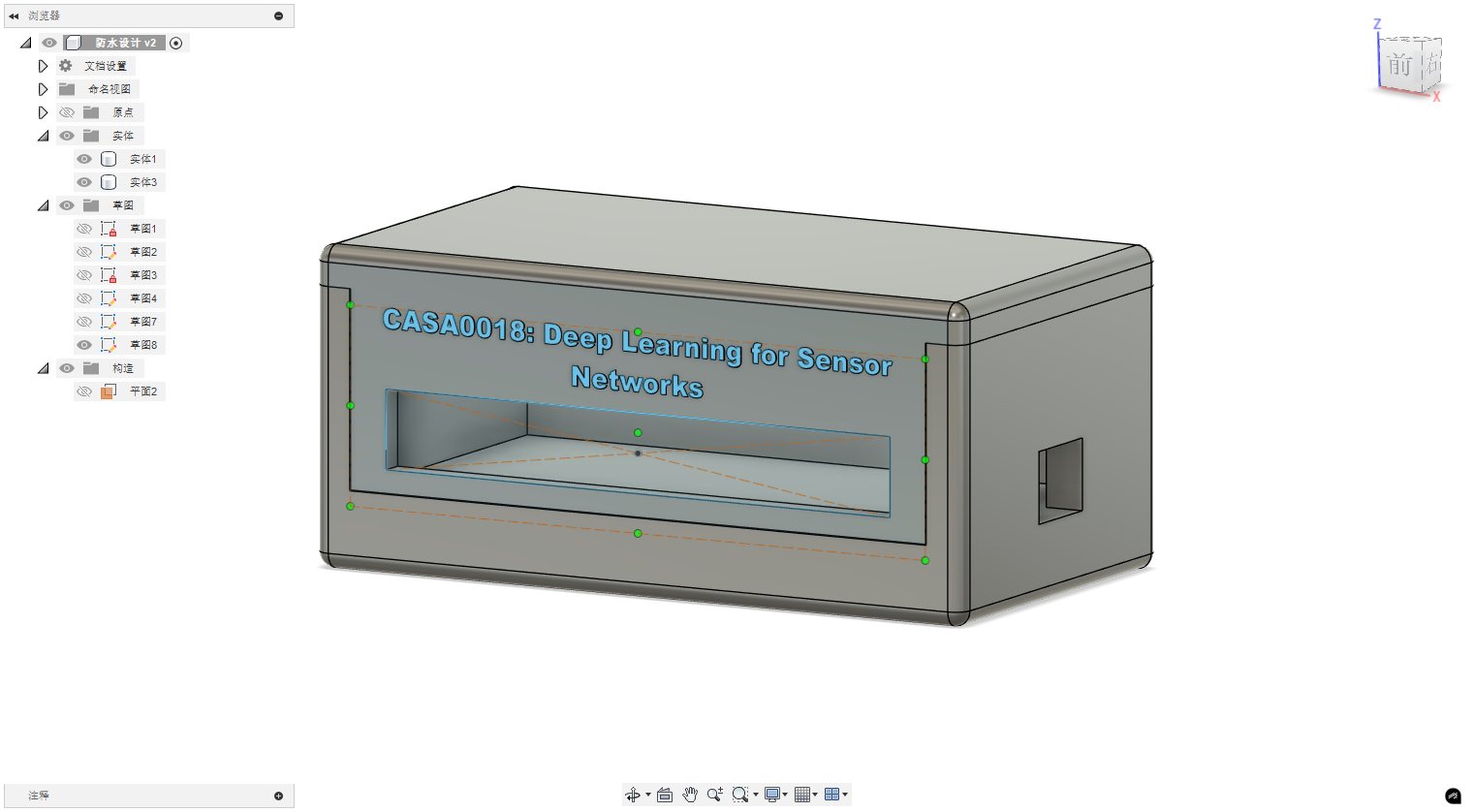
 点底部的 **“Use this model”** 就会把这个作为你当前的“最佳模型”

这个页面我找不的，问问

好像没人知道，我下周查查。

# Week 3

我开始设计防水处理，必须在回国前做出来，我会在下一周给自己放假，因为我要回国了。

设计完了，中间留了3mm的槽我想用来插一个透明的挡板来防水，而接线处的防水我想试试手套收口处的设计，用皮筋和塑料做一个弹性收口。

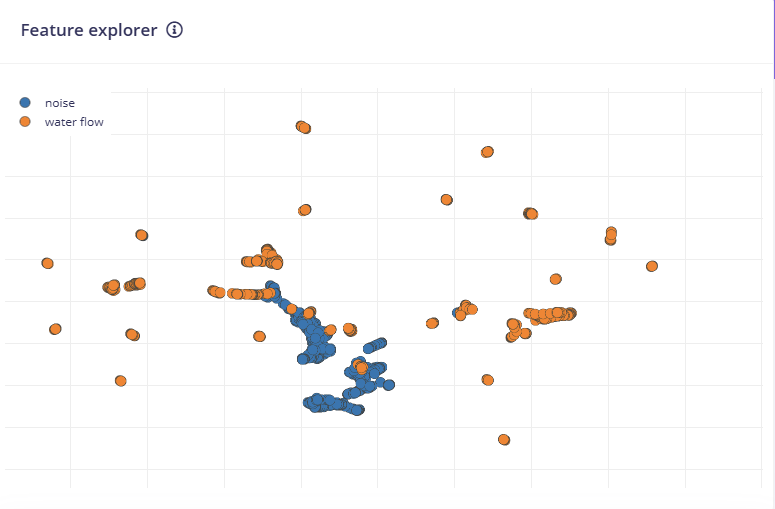
# Week4

收集数据补充到test2，新买的Arduino Nano 33 BLE到了，开始测试防水性同时收集数据，现在我不再在收据数据时可以避开水汽和水雾。

这一周我位于中国，家庭事务繁忙，我无法分出足够的时间用于实验，都是碎片化的时间，要留意事否有错误。

# week5

在中国期间，我将数据集扩展到了16分钟的数据量，



这是现在训练集的 Feature explorer，我复查了在中国整理的数据，Label没有问题。实时证明在套上外壳后，板载的麦克风收音效果出现了变化。为了优化音频切片效果我对MFE 参数进行修改试图优化

1.缩短 Frame Length为0.02

更短的帧长有助于捕捉突发的水声信号变化； Jurafsky & Martin (2021) - Speech and Language Processing

水流声具有高频短时成分，缩短帧长可以提高时间分辨率。 <https://arxiv.org/abs/1807.00129> Sound Event Localization and Detection of Overlapping Sources Using Convolutional Recurrent Neural Networks

2. 适当提升 Filter Number（80）

提高 Mel 频带数，可以让模型更细致地区分水声的频谱形态。 [Sainath et al., 2015] - Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks (ICASSP)

https://arxiv.org/abs/1706.02291 Sound Event Detection Using Spatial Features and Convolutional Recurrent Neural Network

3. 扩大频率范围（尝试 Low = 100, High = 8000）

有些细微的高频“哗哗声”或管道漏水声可能超出 8kHz； <https://arxiv.org/abs/1609.04243> [Choi et al., 2017] - Convolutional recurrent neural networks for music classification

增大频率范围可能增强“水声”检测能力，尤其是在静音环境中。 <https://ieeexplore.ieee.org/document/7952132> [Hershey et al., 2017] - CNN architectures for large-scale audio classification (Google AudioSet)（ Convolutional Neural Networks (CNNs) have proven very effective in image classification and show promise for audio. We use various CNN architectures to classify the soundtracks of a dataset of 70M training videos (5.24 million hours) with 30,871 video-level labels. We examine fully connected Deep Neural Networks (DNNs), AlexNet [1], VGG [2], Inception [3], and ResNet [4]. We investigate varying the size of both training set and label vocabulary, finding that analogs of the CNNs used in image classification do well on our audio classification task, and larger training and label sets help up to a point. A model using embeddings from these classifiers does much better than raw features on the Audio Set [5] Acoustic Event Detection (AED) classification task. ）

4. FFT Length 可以保持 512

如果你将频率范围扩大，可以尝试 FFT length = 1024 以提升频率精度，但代价是计算量提升。 FFT 长度越大，频率分辨率越高（Δf = fs/N）；

如果要精细分辨 200Hz vs 300Hz 的水流频段差异，512 点可能不够；

1024 提供更细粒度，但会增加运算复杂度（重要在 Edge 上部署时权衡）。

<https://www.karolpiczak.com/papers/Piczak2015-ESC-Dataset.pdf> [Piczak, 2015] - ESC: Dataset for Environmental Sound Classification

5. 数据增强：

启用：

Time shift（声音偏移）

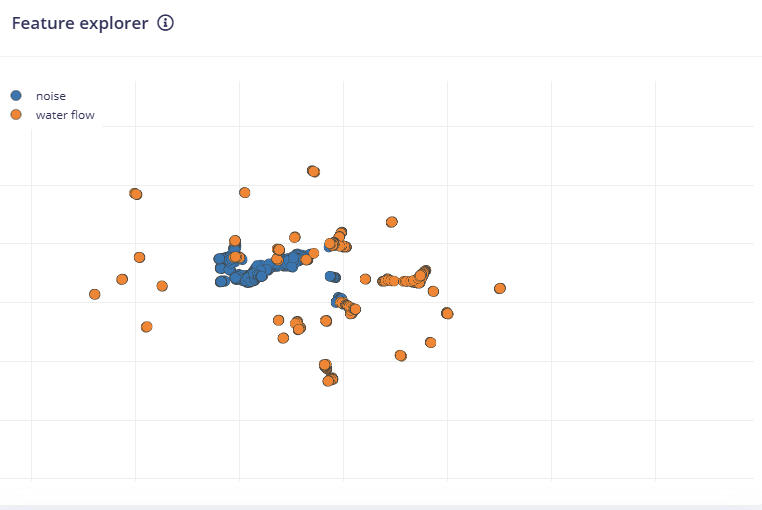
Add noise（白噪声注入）

Time masking / Frequency masking（防过拟合）

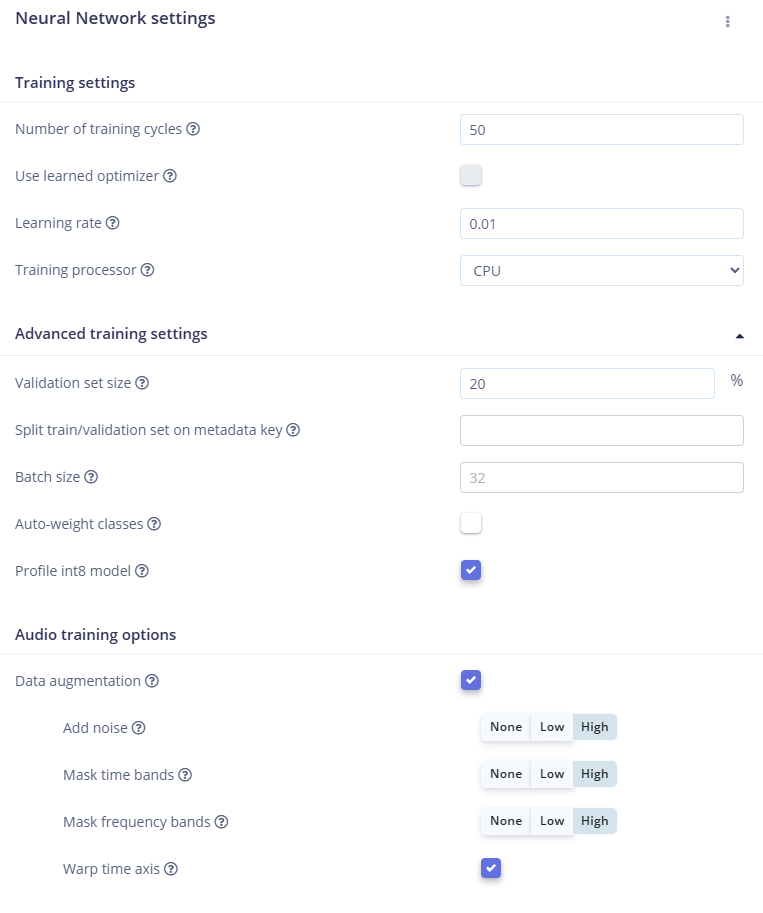
可以让模型更鲁棒地识别不同时段的水流特征。

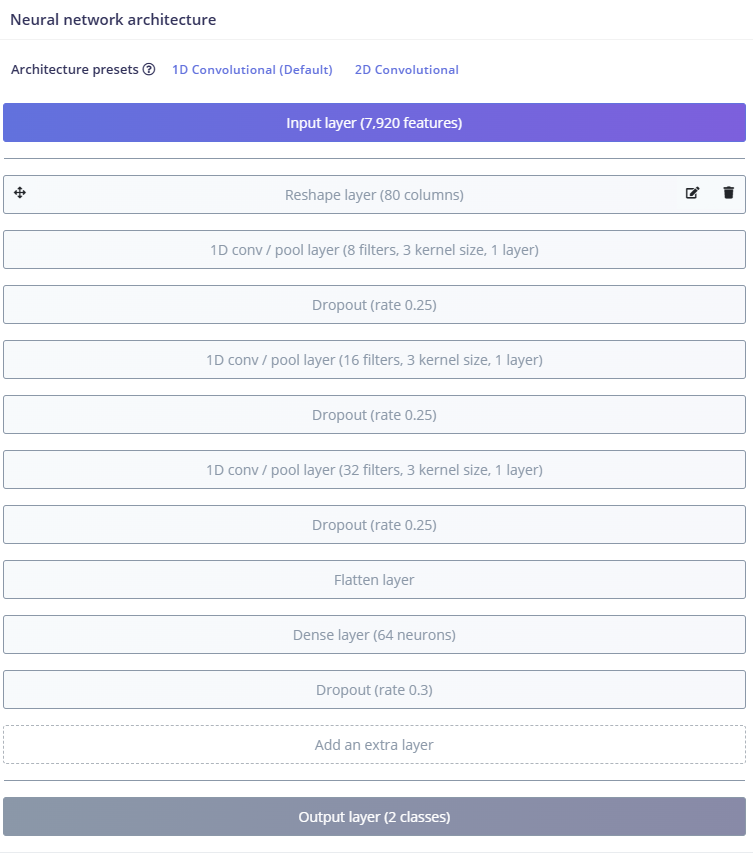
[Park et al., 2019] - SpecAugment: A simple data augmentation method for automatic speech recognition（Google Brain）

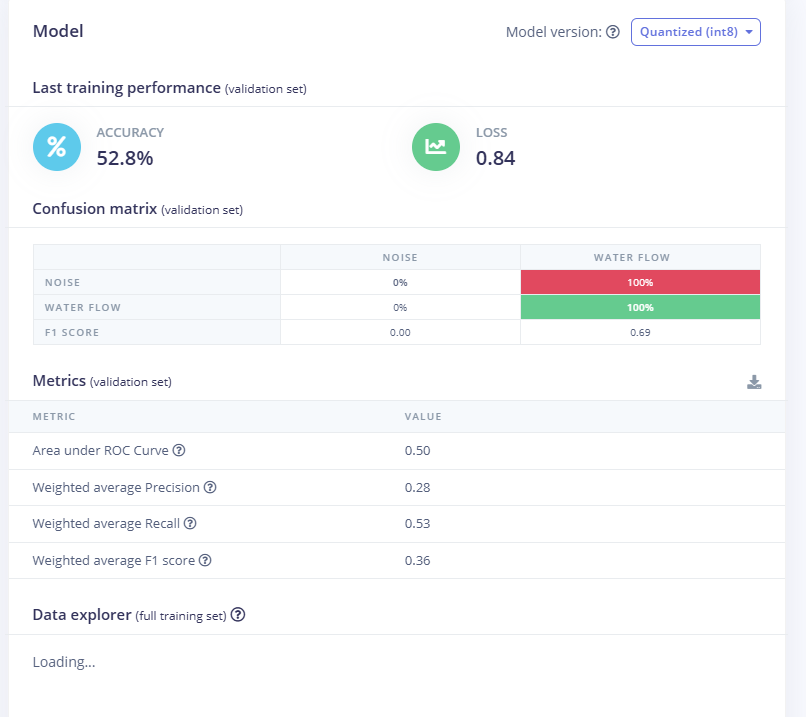
https://ieeexplore.ieee.org/document/7829341 [Salamon & Bello, 2017] - Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification



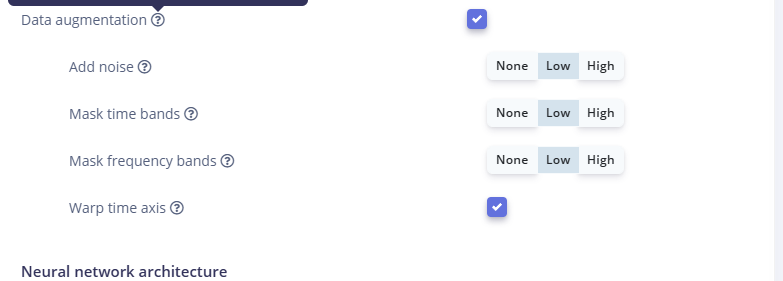
现在Feature explorer变这样了，我不确定这样的特征点是否有优化，我打算直接进行训练然后测试。

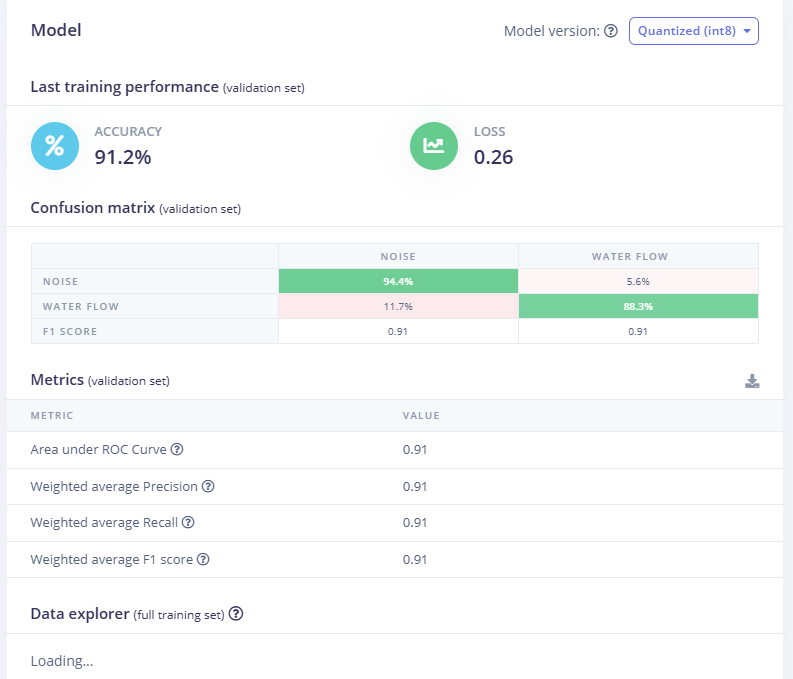




classifier的部分先不做改动。开始训练、

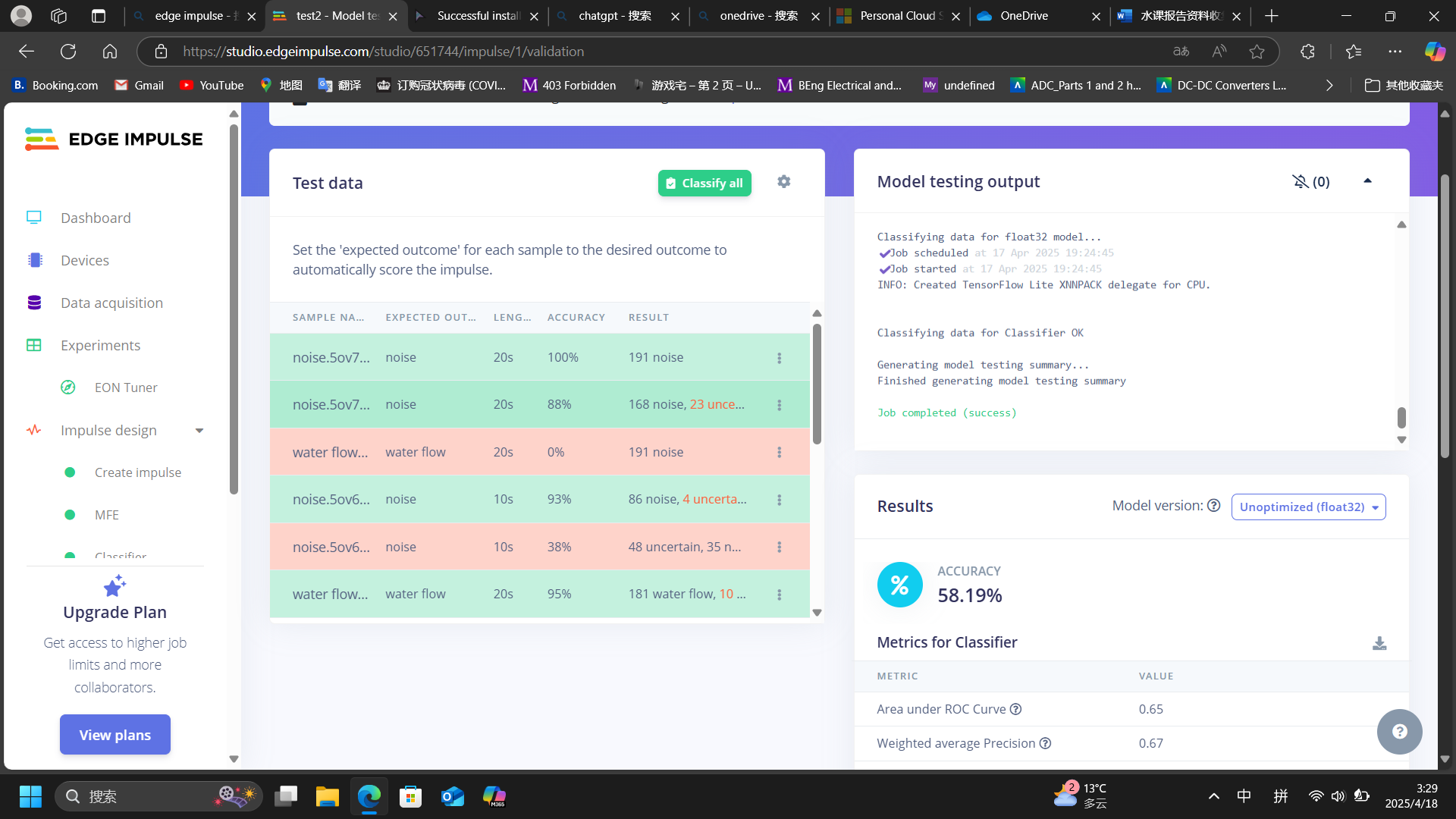
正确率很低，非常不好。我认为可能是数据增强设置过强 → 扰乱了水流声特征

开启了 High 等级的噪声、时间遮蔽、频率遮蔽、时间扭曲，对于“水流声 vs 噪音”这种微弱差异任务，增强过强可能使模型难以学习本来的特征。所以对设置做以下更改



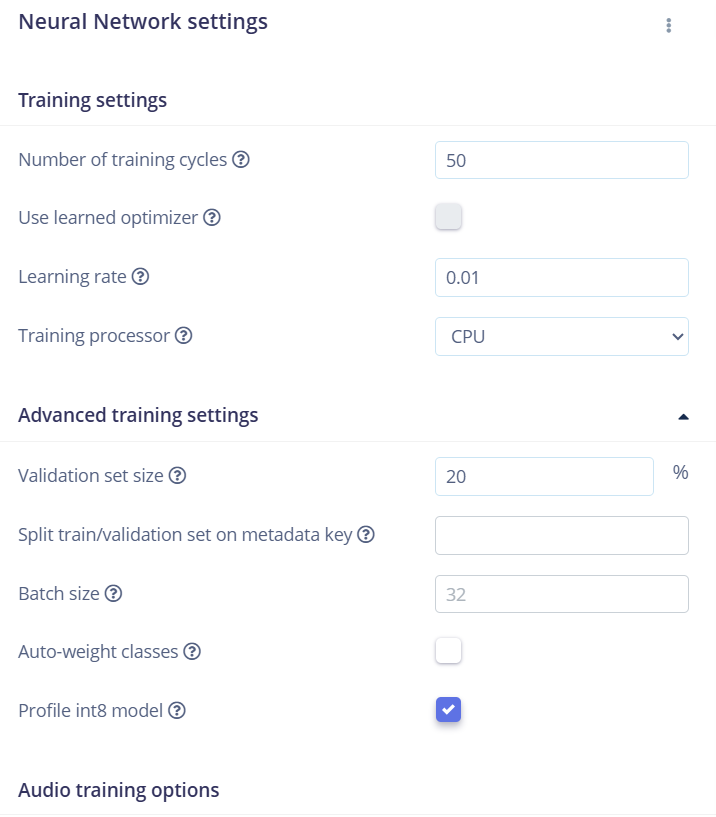
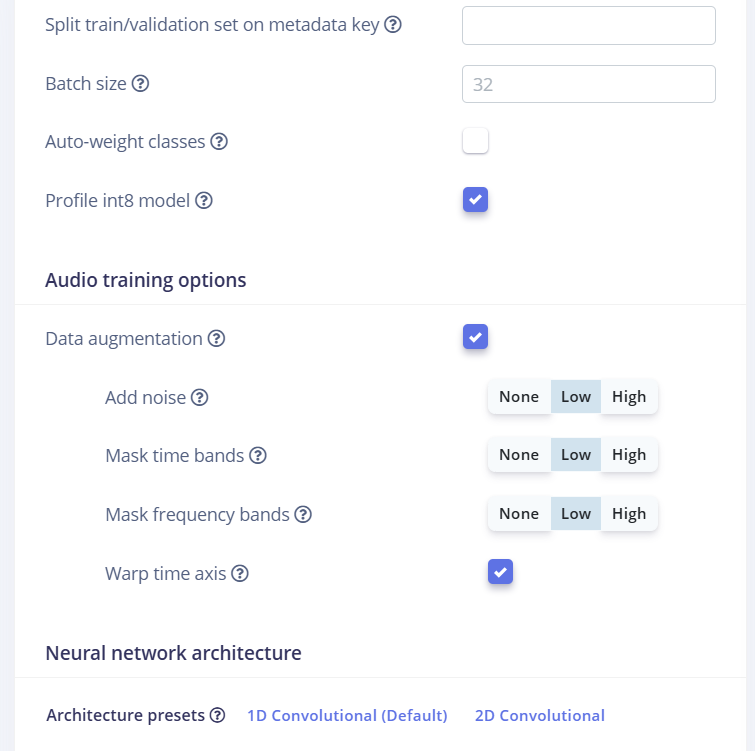
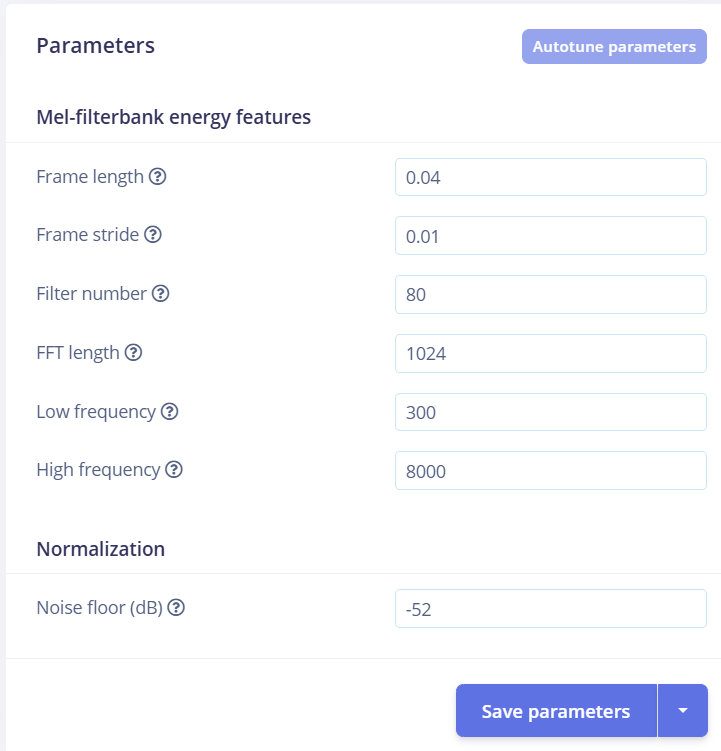
情况有所好转，但是根据 Training output中的内容，epoch15之后loss就开始增加，这让我困惑，为什么数据量增加了overfiting的到来却提早了呢？

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Train Accuracy** | **Val Accuracy** | **Val Loss** | **评论** |
| 3 | 93.1% | 70.5% | 0.47 | 起步不错 ✅ |
| 5 | 94.1% | **85.1%** | 0.38 | 初步提升 ✅ |
| 9 | 95.0% | **86.4%** | 0.30 | 很棒，模型开始泛化 ✅ |
| **15** | **95.9%** | **91.4%** | **0.26** | 🎯 **最佳点**：模型泛化能力最强 |
| 27 | 96.1% | 86.9% | 0.30 | 依然优秀 |
| 50 | 96.6% | 70.6% | 0.75 | 已明显过拟合 ⚠️ |

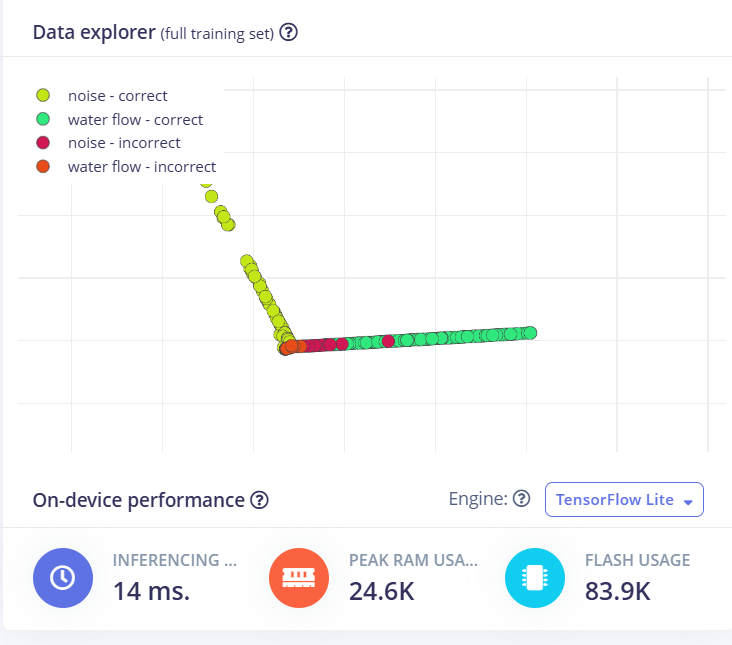




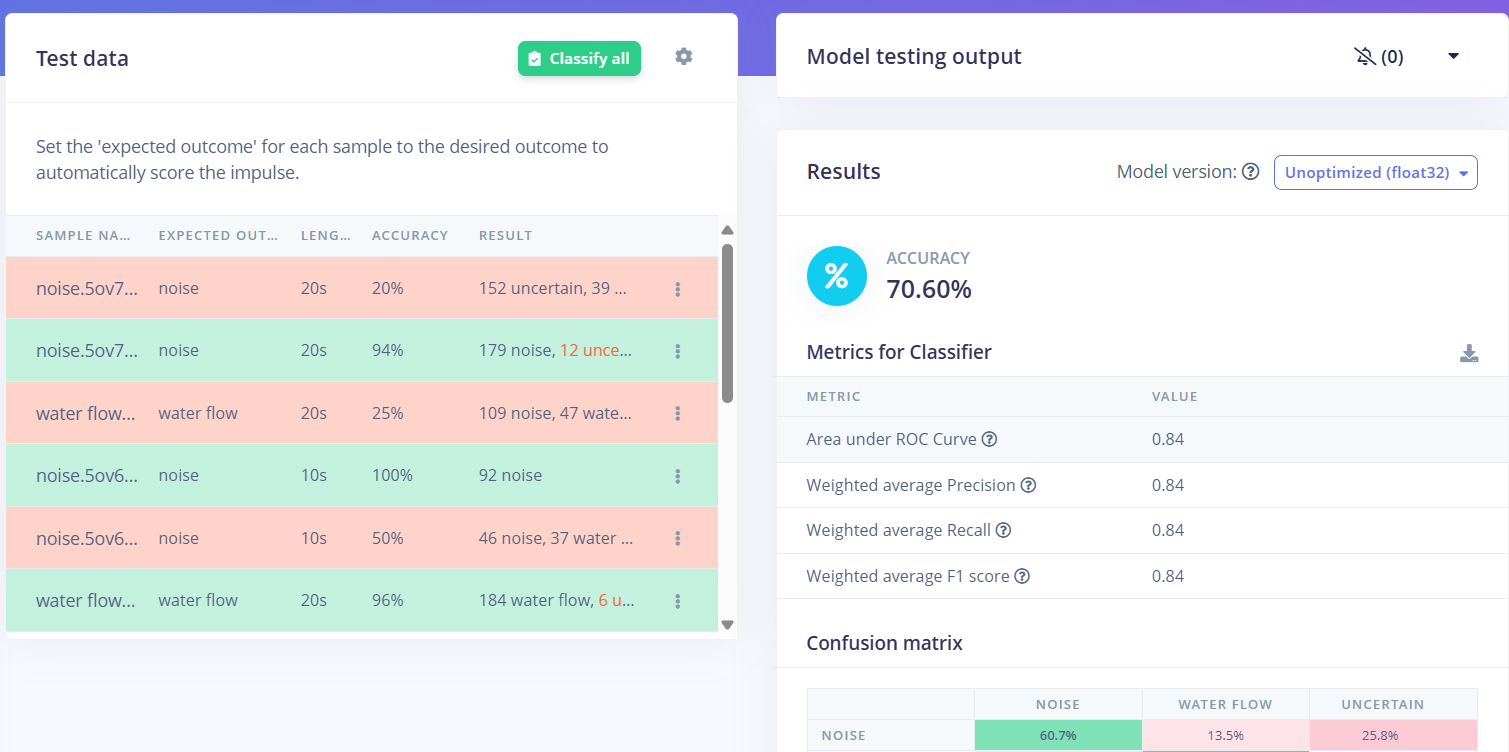
训练结果不尽如人意,我调整Epochs再训一次，如果失败我把 Frame length改回0.04再切片训练集一次。最终我对MFE页面做了以下更改：



得到以下结果



其在测试集中的表现如下：70.60%的正确率，也可以接受，但明显没有上一次的好，我增加数据集和测试集并不应该得到这个结果。



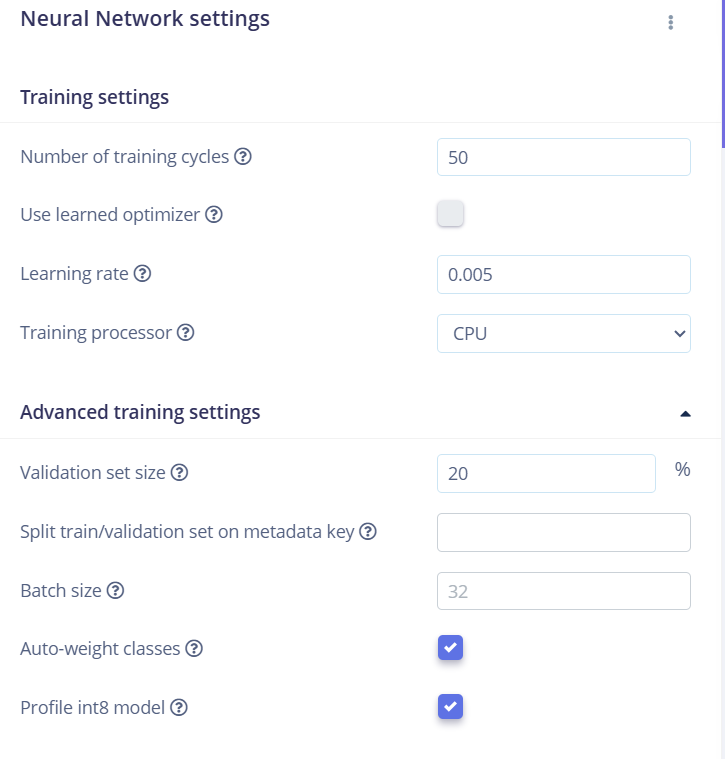
我试图通过降低学习率但增加epochs的方式来避免模型的过拟合，但edge impulse并不支持超过20分钟的训练量。

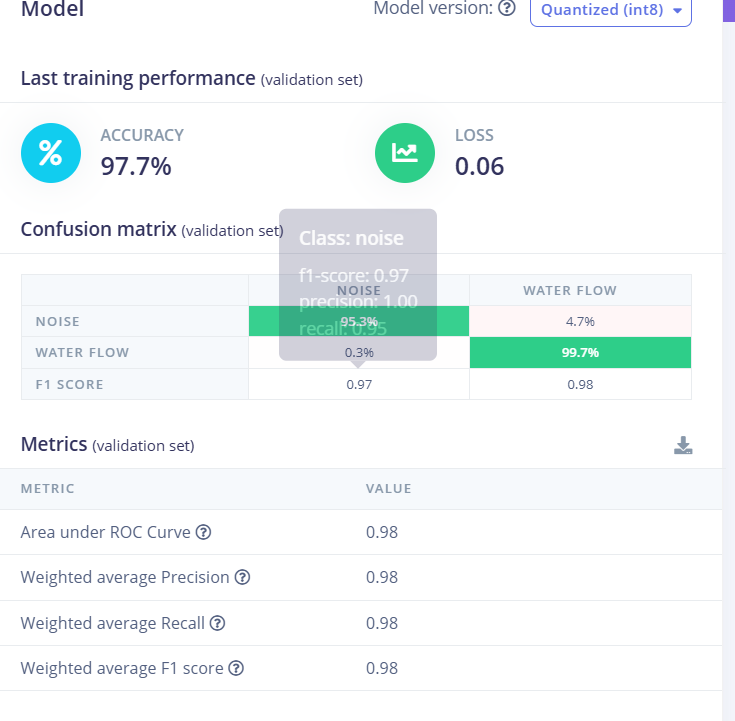
ERR: Estimated training time (26m 47s) is larger than compute time limit (20m 0s).

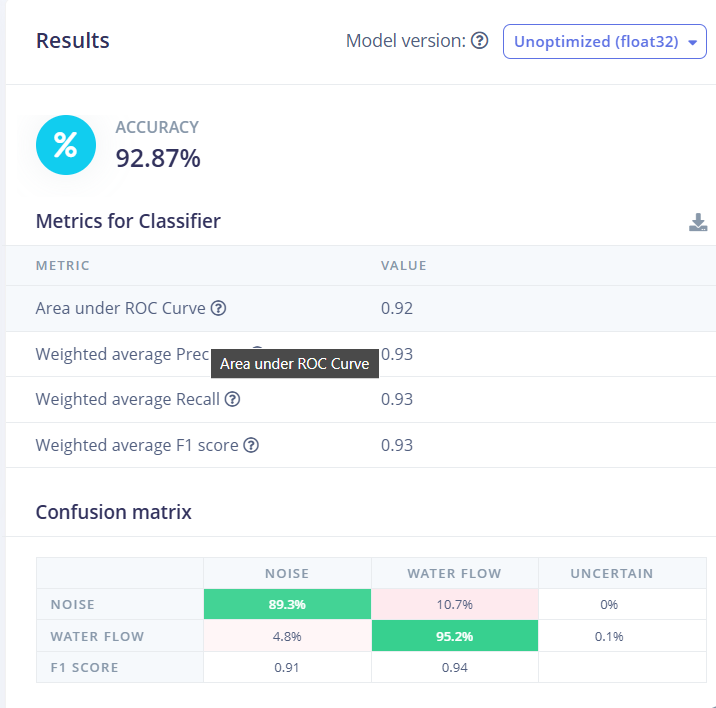
See https://docs.edgeimpulse.com/docs/tips-and-tricks/lower-compute-time on tips to lower your compute time requirements.

Alternatively, the enterprise version of Edge Impulse has no limits, see https://www.edgeimpulse.com/pricing for more information.

Application exited with code 1

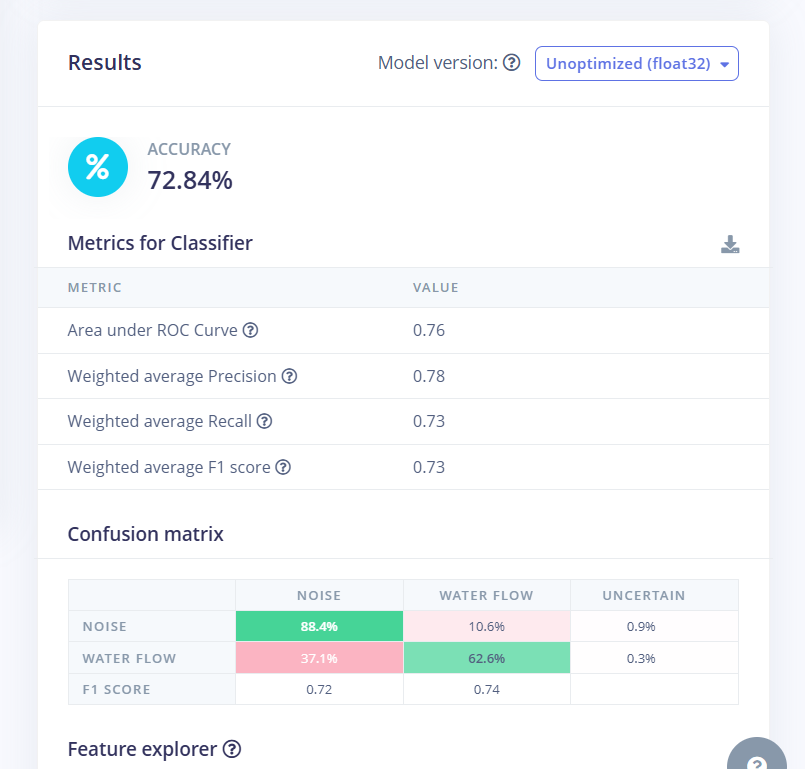
Job failed (see above)  
所以在有限的资源下，我只能选择将加强学习弱化，用已有的训练并找到最合适的epochs，顺便，我把采样的window side改为2秒了，因为我认为识别没关闭的水流就算休要使用两秒的时间也合理。考虑到模型大小和切片时长，我把FFT Length也降低到512了，主要是为了节省时间。但是在此时也让我有了一种猜测，有没有可能是数据加强了之后的训练集难以找到特征，而edge impulse在训练时长上的上限使得我无法继续加大epoch用一种简单的方式的得到更复杂的模型。所以我索性关闭了加强学习，单纯使用以下数据进行训练：

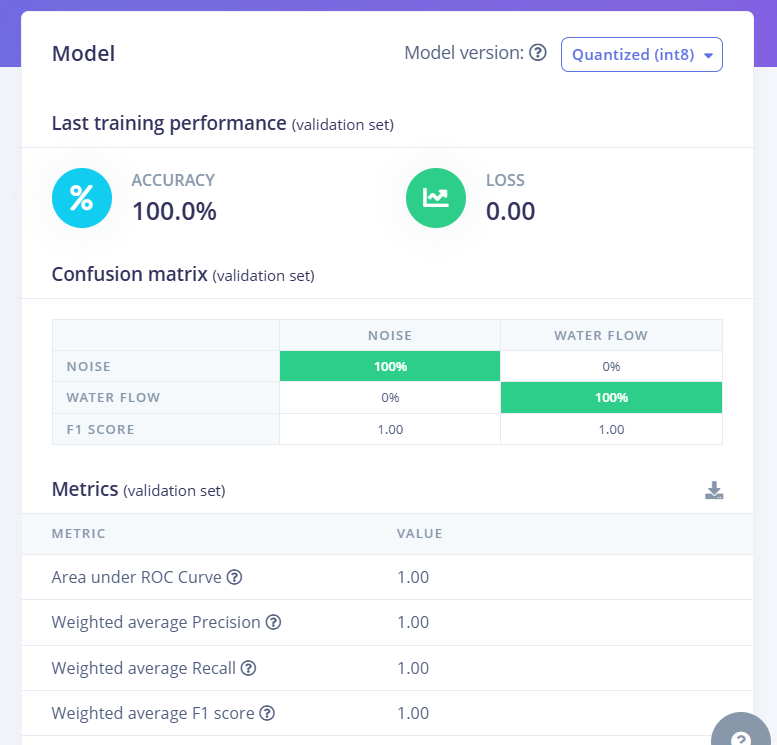
得到以下结果：

而在测试集中，它的表现也非常完美

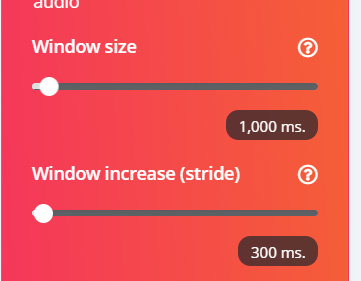
Window side好像只能是1000，我用了2000ms跑出来的模型不能在BLE33上运行，使用的是 EI\_CLASSIFIER\_SLICES\_PER\_MODEL\_WINDOW = 4（每个模型窗口分成 4 片）

每片帧（slice）需要的样本数较高，如果板子来不及采满一整帧，就会导致出错！，我想直接修改 EI\_CLASSIFIER\_SLICES\_PER\_MODEL\_WINDOW = 4代码为1，但是好像不行。这版的模型文件叫 ei-test2-arduino-1.0.5和 ei-test2-arduino-1.0.6。

ei-test2-arduino-1.0.7是一版本训练集正确率高达百分之百的奇特模型，在测试集中检测发现，只要有干扰，它的检测能力就会显著下降，鲁棒性非常低。



在考虑到部署时遇到的问题，我将window size和window increase改回官方教程中的数值。由此得到的 ei-test2-arduino-1.0.8.zip文件也是一个非常精准的模型，但是还是鲁棒性不足，对于雨声任然会识别成水流声，在有大量背景噪音的情况下对水流声也很难分辨。



对ei-test2-arduino-1.0.7.zip的部署试验结束了，在理想环境下可以运行，也有一定的抗干扰能力，但是对于较小水流声并没有检测能力

noise: 0.99609

water flow: 0.00391

Predictions (DSP: 108 ms., Classification: 39 ms., Anomaly: 0 ms.):

noise: 0.99609

water flow: 0.00000

Predictions (DSP: 107 ms., Classification: 39 ms., Anomaly: 0 ms.):

noise: 0.99609

water flow: 0.00000  
这是部分取样效果，以此推断将启动蜂鸣器的阈值设置为water flow:0.9是合理的。

我整合了蜂鸣器代码，这个文件目前在 test2\_inferencing1.0.7非压缩文件夹中。实验测试结果，在安静理想的环境下工作良好，这个仪器甚至可以在距离水源一米开外的距离检测是否有水流流动的声音，考虑到deep learning的不稳定性，我将触发蜂鸣器的阈值设置的很高为：0.9f来避免繁杂的背景噪音误触发，这使得如果是较弱的水流，或者有环境音干扰，这个检测会非常不稳定，但这刚好在弱水流的情况下带来了短促触发的效果，而当水流明显强烈，蜂鸣器连续触发会将效果变为尖锐长鸣。而在长时间稳定性测试中，我也发现一些问题，比如在强烈的水流连续数秒触发water flow标签后，water flow的label的输出值有可能会保留在0.9以上，，然后在很长一段时间后突然恢复正常，或者被我强行断电。可能是一直触发的蜂鸣器作为背景噪音干扰了模型的判断导致其一直误判还有水流声，这说明如果模型的准确率无法继续提升，将蜂鸣器作为提醒设备的设计必须改变，用蓝牙发送至手机水没关的消息或许更好。但在目前来说，目前模型在安静环境或仅人声环境中工作良好。

总之，目前的模型和部署测率只是提供了一种开发方向，基于已有的模型和准确率，为了可以轻易做到多方面拓展。比如放弃使用板载麦克风而使用外接的高品质麦克风来提高可接收音频的品质，或者使用在代码上加入滑动窗口 加 连续命中判定，只有当“目标标签”在**连续 N 次推理中都高于阈值**，才触发蜂鸣器，但这无疑会加大在小水流或者噪音环境中对蜂鸣器的触发难度。

在有“周期性噪声”或频繁干扰的情况下，可以使用抖动抑制方案的代码，也就是“蜂鸣器响一次后，需要过一段时间才能再次响”，但这会使得原本设计的连续长鸣功能不可用。

而在模型上的优化，可以融合多个标签概率，将原本单一的“water flow”,分类为“tap”“shower”“flush”等，组合这些概率判断。融合多个标签的概率可以显著提升模型在实际场景下的判断准确率，因为现实中的声音往往具有模糊性和连续性，一个声音事件可能同时激发模型对多个相关标签的响应。例如，水流声可能同时被模型部分识别为“tap”、“shower”或“flow”。如果仅依赖单一标签的高置信度作为触发依据，容易导致漏检；而将多个标签的置信度叠加并综合判断，则能更好地容纳模型的不确定性，反映更接近人类直觉的判断方式。这种方法在提高召回率的同时，也提升了系统对真实复杂环境的适应能力。

待办事项：**我觉得每个合格的模型都可以使用classification和transfer learning两种方法训练，但是至少在model testing正确率高于百分之80之后，并找到最好的epoch，在进行对比，不然没有意义。同时，classification的不同层架构直接也应该进行对比。**

**在这个阶段我也认为我应该设置一个合理的模型合格标准了，我认为初步及格应该是modeling teat正确率高于80%，第二步是通过那个低噪点噪音测试。第三步是封装后的长时间实际测试。**

**做防水处理，不然收集不到有水雾状态的数据。可以用PLA打印两块镂空的板子，中间夹上保鲜膜，连接处使用榫卯结构。**