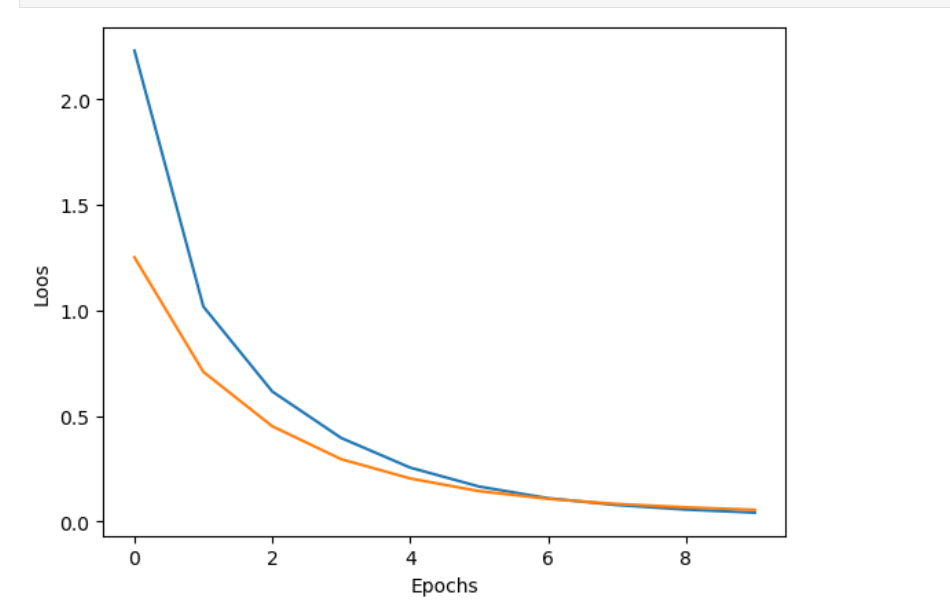
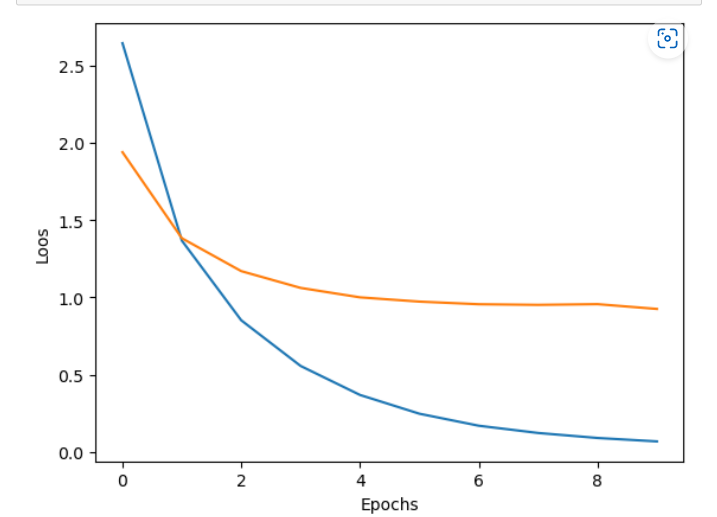
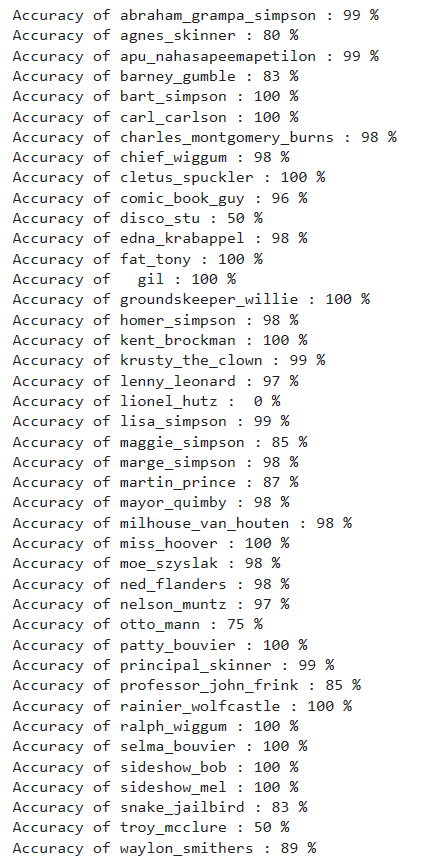
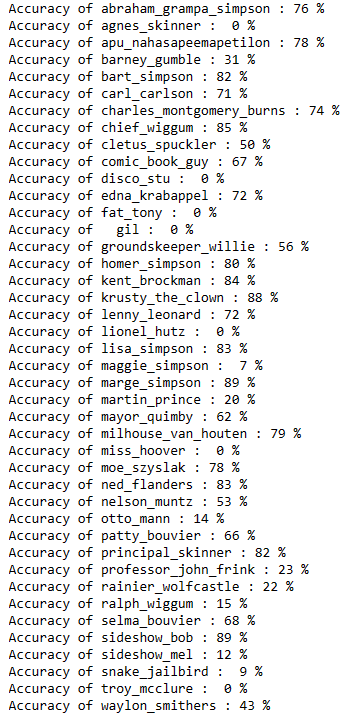
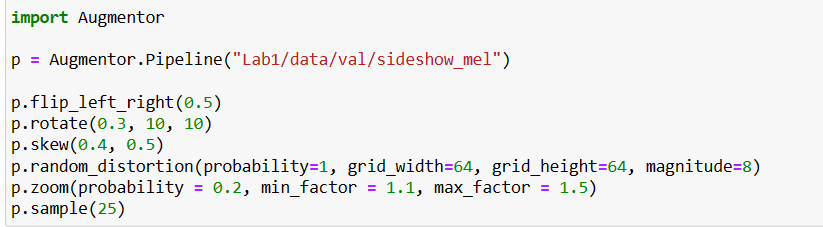
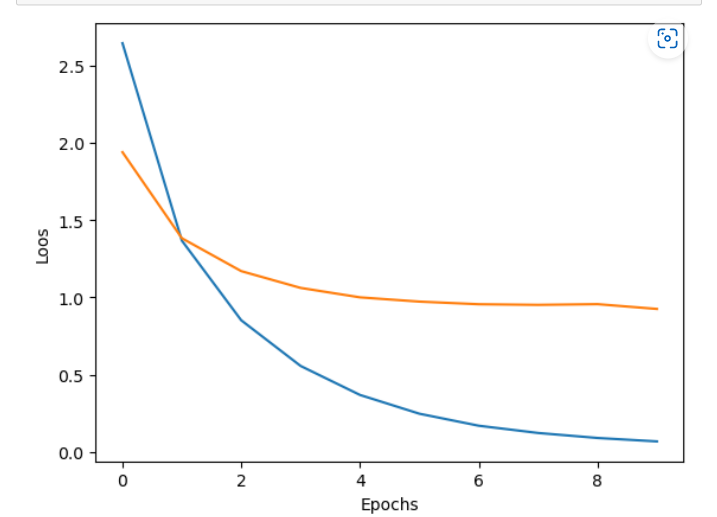
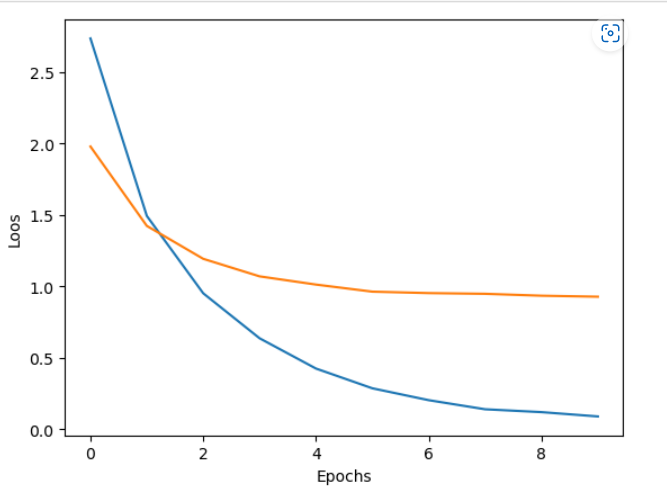
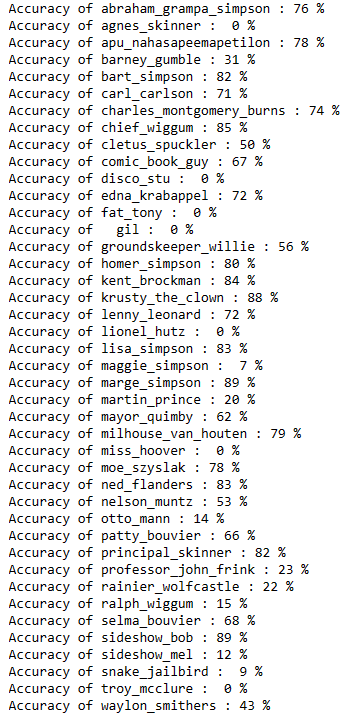
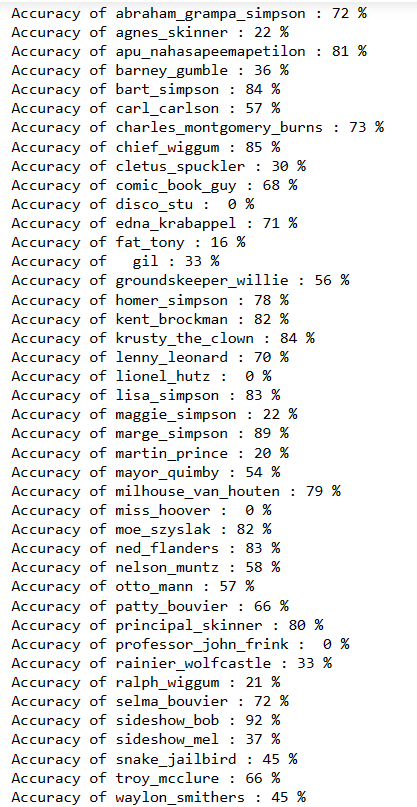
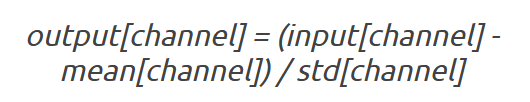
1. Перегрузила датасет, чтобы данные с train не дублировались в val и test
2. Точность значительно снизилась  
     
    до после  
    
3. Использовала аугментацию, чтобы забалансить датасет, а именно часть train  
    Примеры:  
      
      
   Код для аугментации:  
   



1. Снова школа:  
   до после   
    
2. Добавила метрику f1 – среднее между precision и recall, тк нам одинаково важно сколько модель нашла неправильных и не нашла правильных..... В общем, прощаем ошибки первого и второго рода в равно мере.
3. Нормализация картинки. Нормализация помогает получить данные в пределах диапазона и уменьшает асимметрию, что помогает учиться быстрее и лучше. Нормализация также может решить проблемы уменьшающихся и взрывающихся градиентов.  
   По формуле можно вычислить наиболее подходящие числа для нормализации  
     
   Стремимся к тому, чтобы среднее входных данных было равно нулю, а стандартное отклонение – единице
4. with torch.no\_grad()  
   Отключение вычисления градиента полезно для логического вывода, когда вы уверены, что не будете вызывать [Tensor.backward()](https://runebook.dev/ru/docs/autograd?page=2" \l "torch.Tensor.backward). Это уменьшит потребление памяти для вычислений  
   requires\_grad=True для вычисления градиента дефолт тру  
   requires\_grad=False экономит память время и ваши нервы