Metric Learning

Домашнее задание выполняли в [jupyter-ноутбуке](https://github.com/kVesnin/computer-vision_homeworks/blob/main/metric_learning/Untitled.ipynb), однако, чтобы было удобнее просматривать код, вынесли его в [python-файл](https://github.com/kVesnin/computer-vision_homeworks/blob/main/metric_learning/experiments.py). Весь код лежит на [github](https://github.com/kVesnin/computer-vision_homeworks/blob/main/metric_learning/).

Здесь мы сначала расскажем о проведённых экспериментах, а затем приведём графики loss-ов и метрики для каждого эксперимента. Код легко воспроизводится - точка входа для каждого эксперимента единая: функция launch\_experiment.

**Что мы попробовали:**

1. **Triplet Margin Loss** - этот loss стремится отдалить эмбеддинг анкоров от эмбеддингов негативов больше, чем от эмбеддингов позитивов, минимум на величину margin. В частности, пробовали выставлять разный margin.
2. **batch-hard семплирование** - в каждом батче для каждого семпла в качестве positive и negative используются самый далёкий пример того же класса и ближайший пример другого класса.
3. **Proxy-Anchor Loss** - каждому классу соответствует эмбеддинг, к которому стремятся все эмбеддинги того же класса (в каждый эмбеддинг мы тоже пробрасываем градиент при обучении). Использовали этот Loss как вспомогательный к Triplet Margin Loss-у, перебирали разные lr для весов эмбеддингов, а также разный коэффициент в loss-е.
4. **KNN + Faiss** - попробовали на этапе инференса не просто усреднять все эмбеддинги одного класса и брать ближайший, а реализовать другой подход. Все эмбеддинги для семплов из train-датасета мы положили в faiss-индекс, а на этапе валидации находили top-k ближайших эмбеддингов. Затем смотрели на их метки классов и голосованием определяли, к какому классу определить текущий семпл.

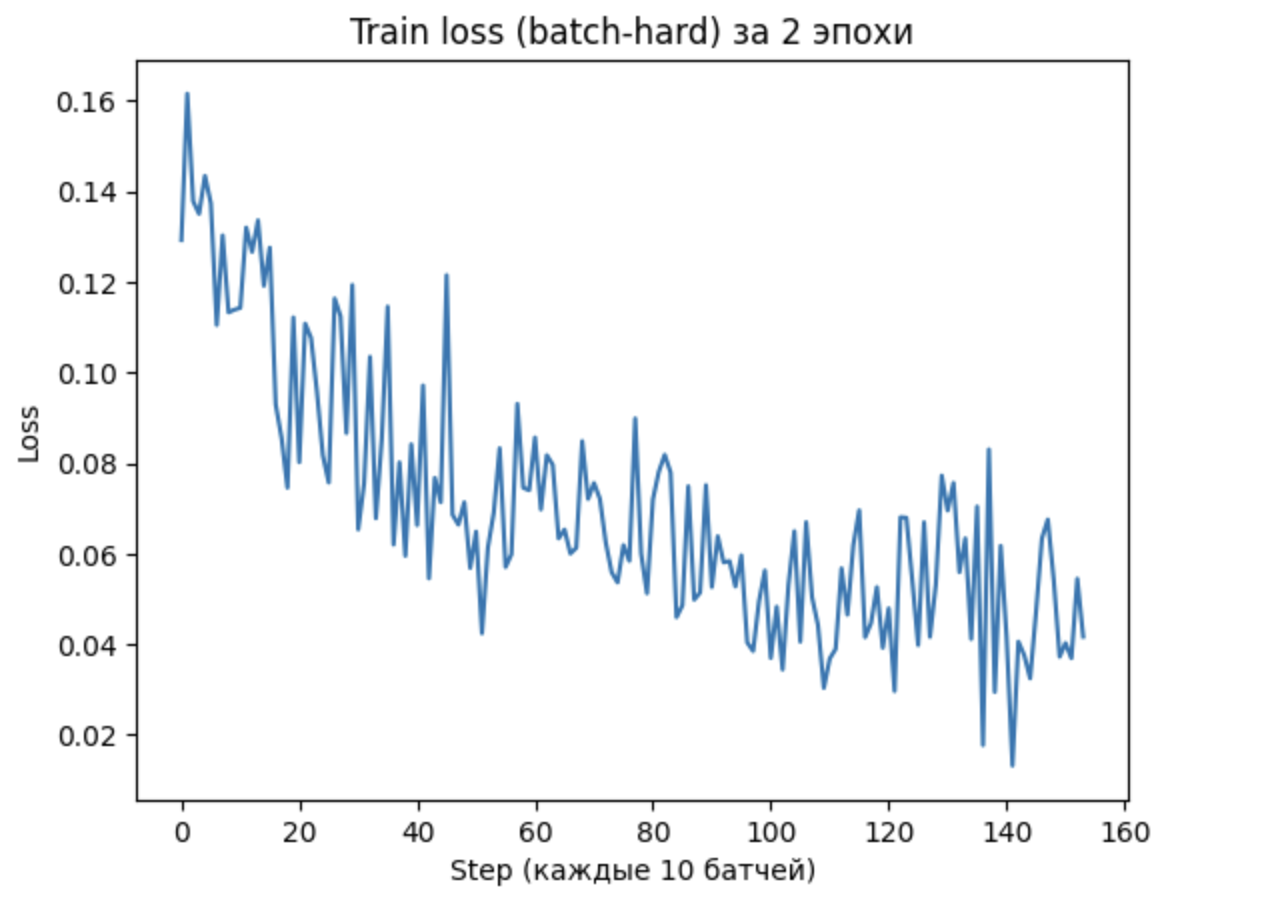
# Графики и метрики

1. **Triplet Margin Loss, margin = 1.0, semi-hard семплирование**

****

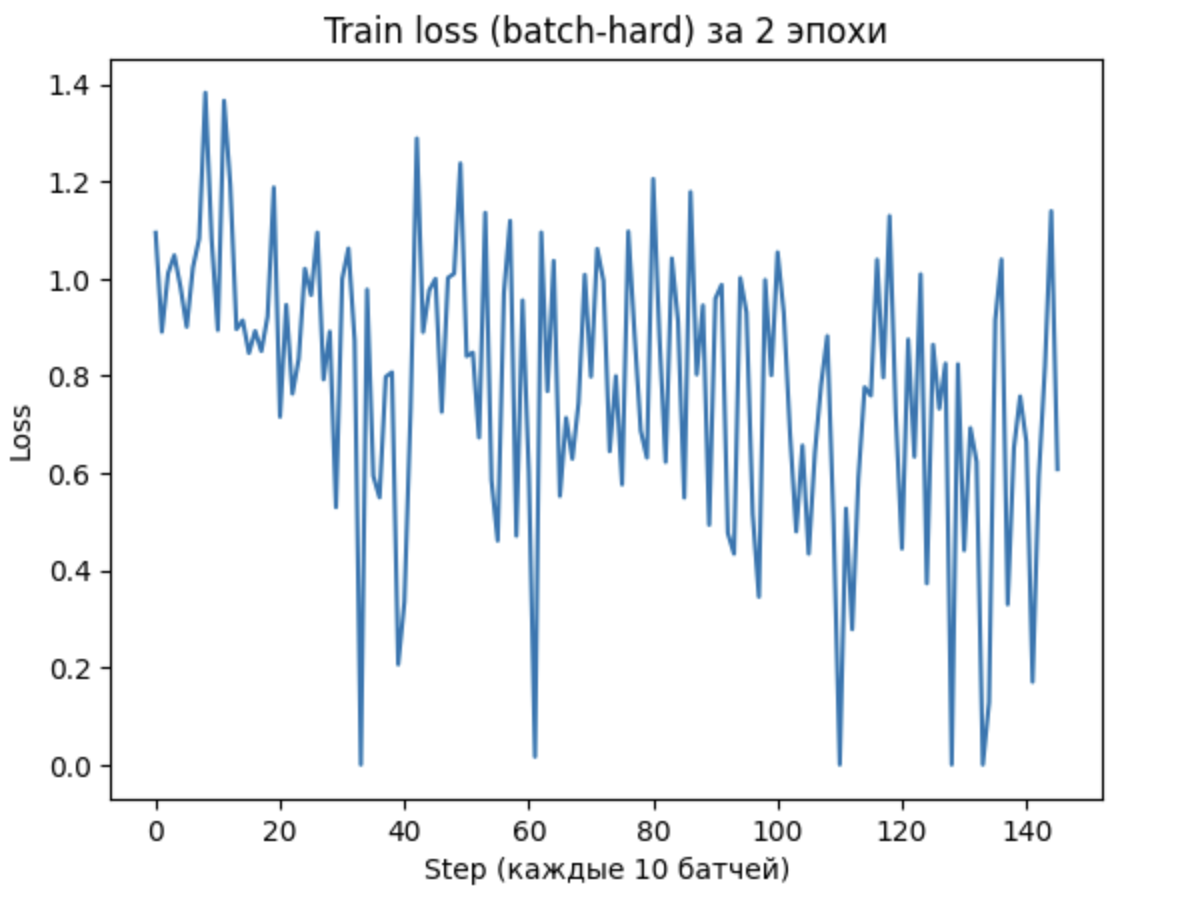
**Accuracy = 0.8051**

1. **Triplet Margin Loss, margin = 0.2, semi-hard семплирование**

****

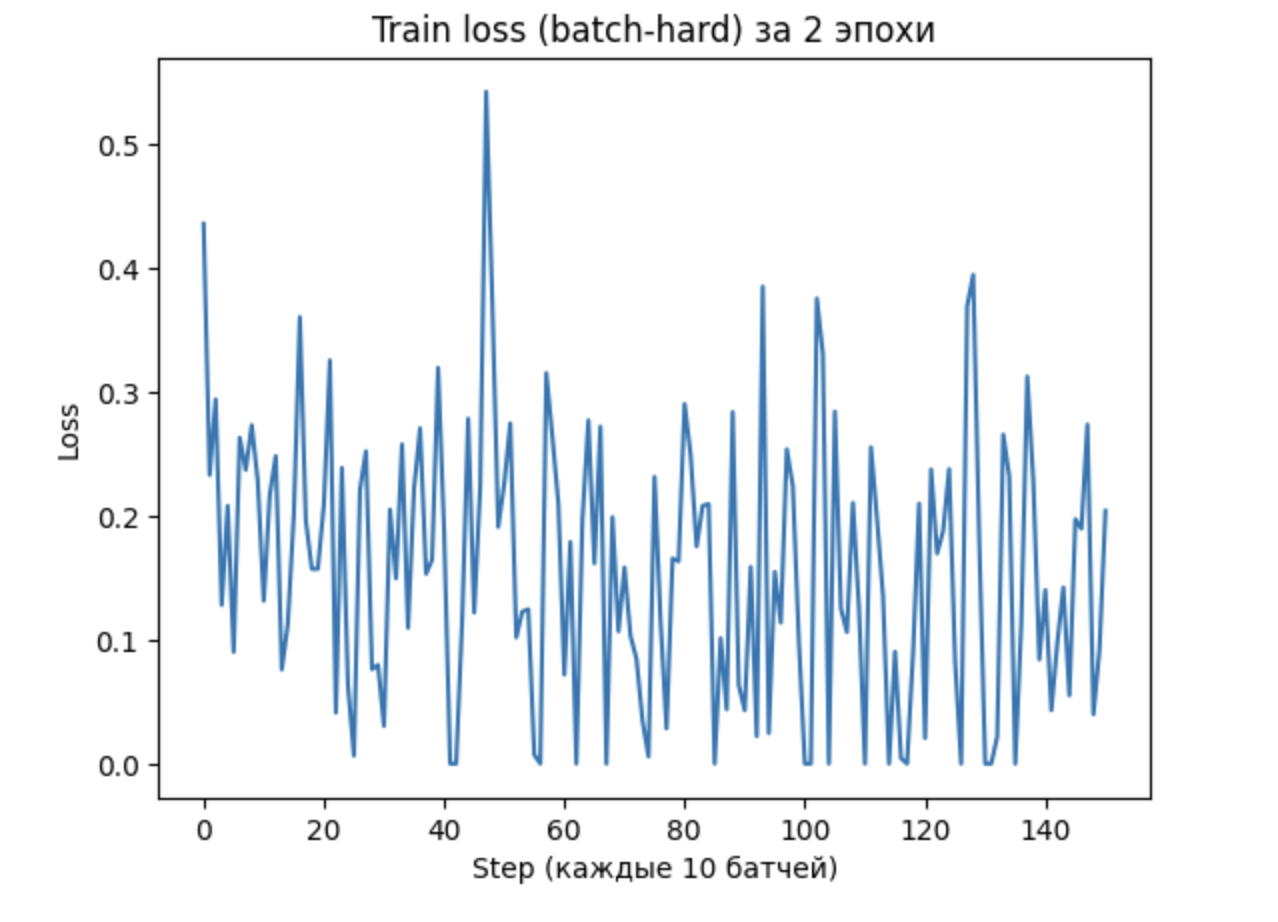
**Accuracy = 0.8120**

1. **Triplet Margin Loss, margin = 1.0, batch-hard семплирование**

****

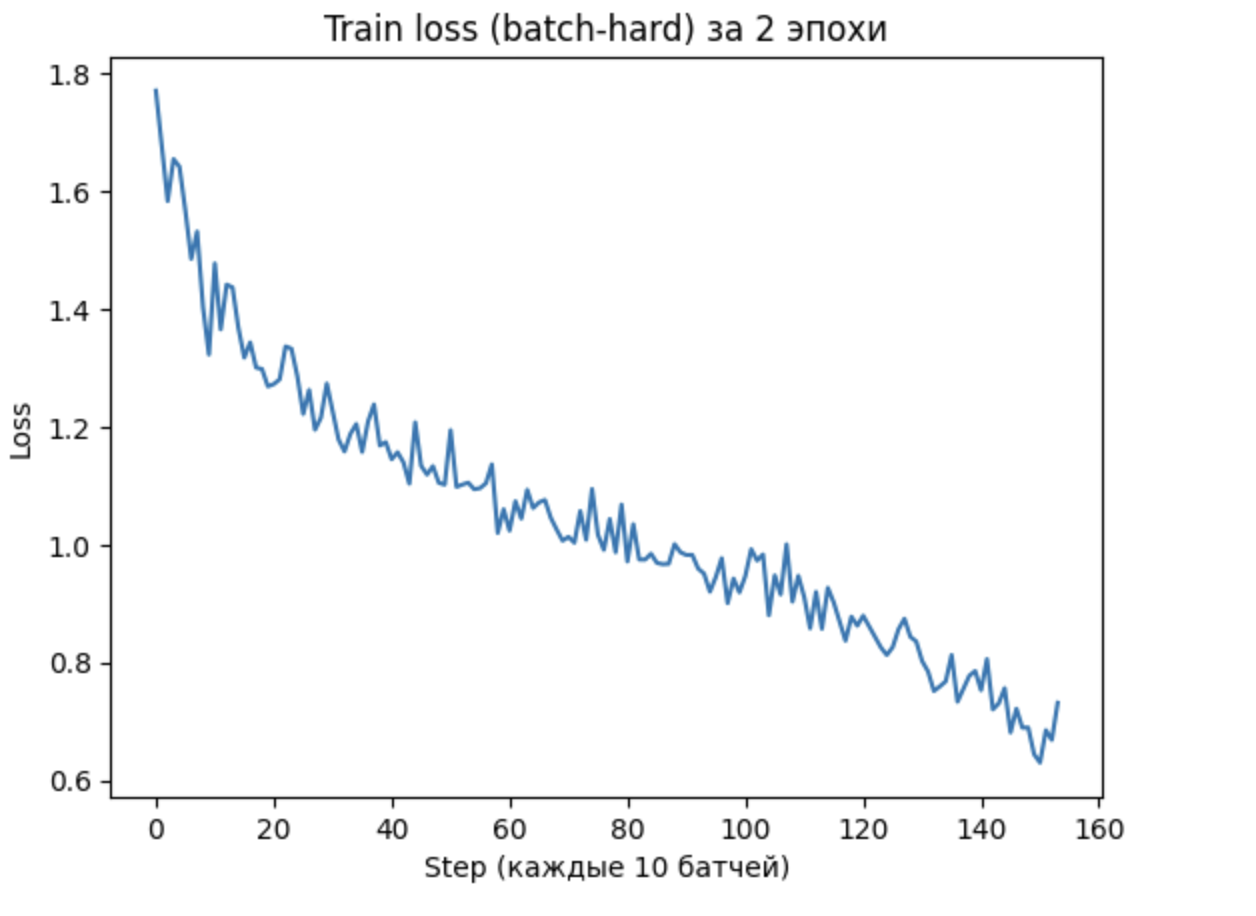
**Accuracy = 0.6986**

1. **Triplet Margin Loss, margin = 0.2, batch-hard семплирование**

****

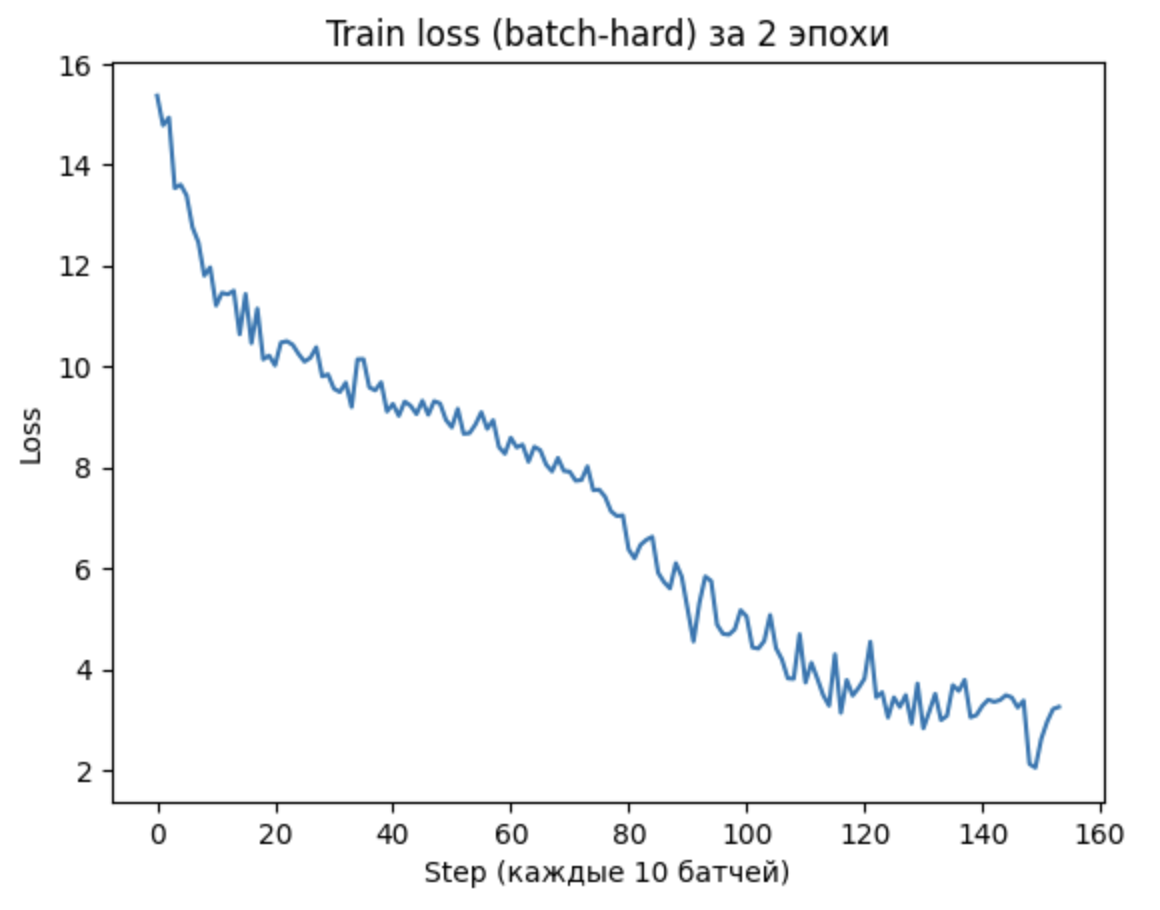
**Accuracy = 0.7429**

1. **Triplet Margin Loss + Proxy Loss, margin = 0.2, semi-hard семплирование, proxy coef = 0.1, proxy\_lr = 1e-4**

****

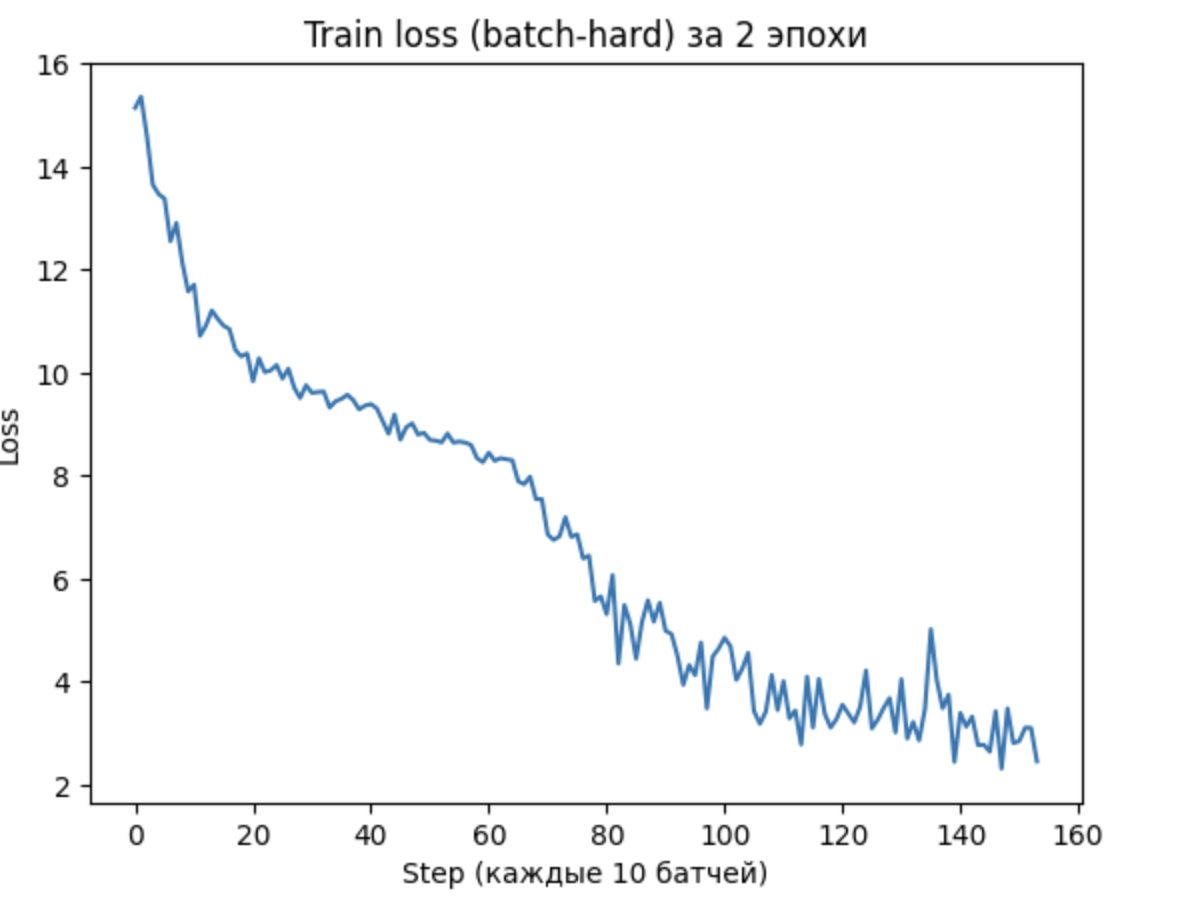
**Accuracy = 0.7988**

1. **Triplet Margin Loss + Proxy Loss, margin = 0.2, semi-hard семплирование, proxy coef = 1.0, proxy\_lr = 1e-4**

****

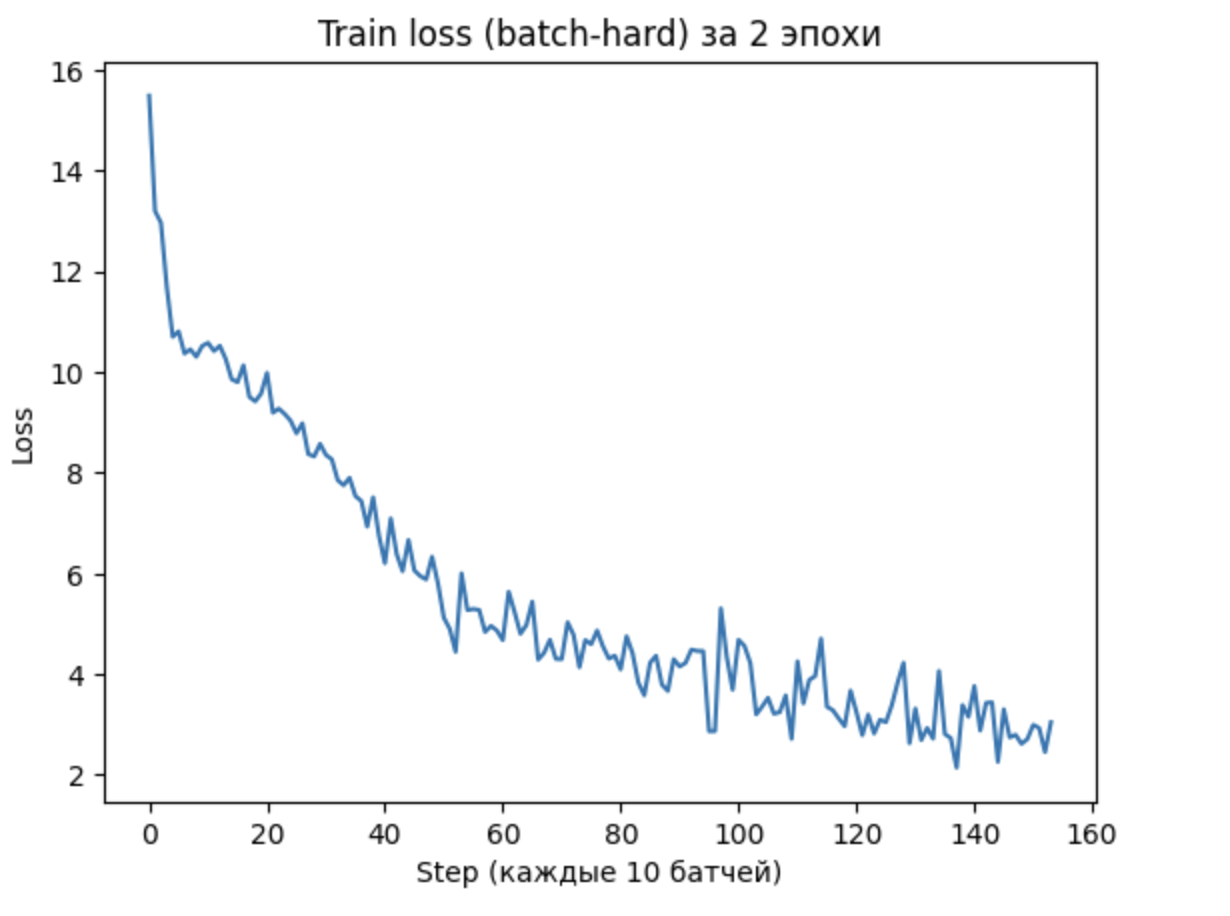
**Accuracy = 0.7881**

1. **Triplet Margin Loss + Proxy Loss, margin = 0.2, semi-hard семплирование, proxy coef = 1.0, proxy\_lr = 1e-3**

****

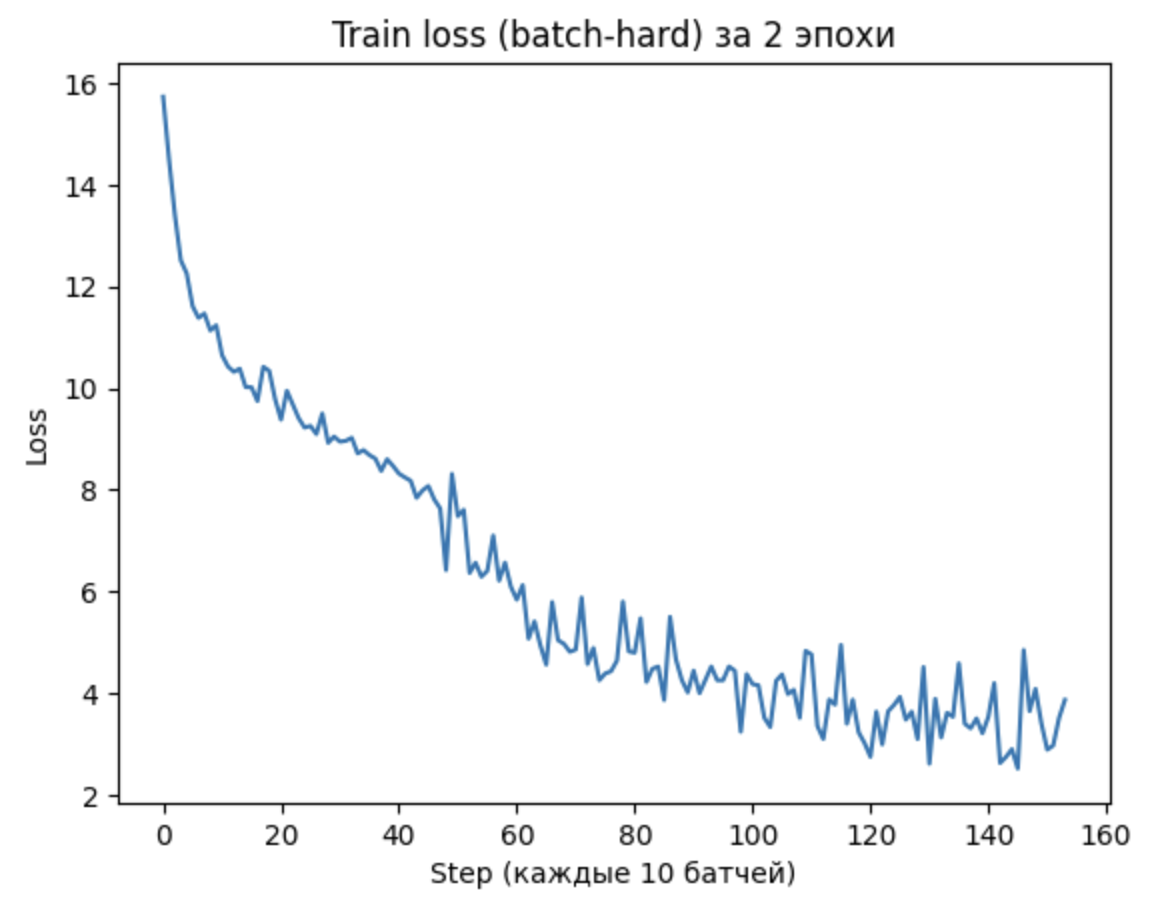
**Accuracy = 0.8229**

1. **Triplet Margin Loss + Proxy Loss, margin = 0.2, semi-hard семплирование, proxy coef = 1.0, proxy\_lr = 1e-2**

****

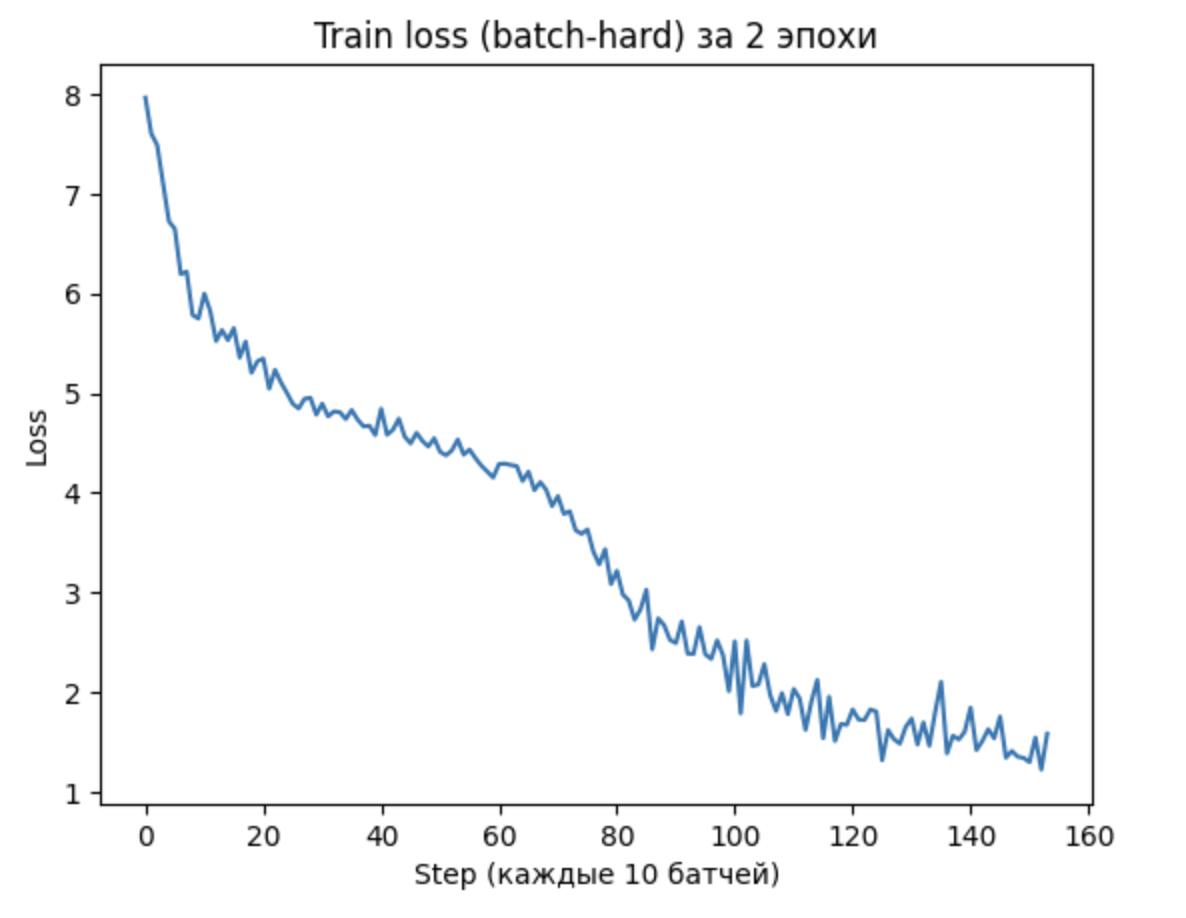
**Accuracy = 0.8136**

1. **Triplet Margin Loss + Proxy Loss, margin = 0.2, semi-hard семплирование, proxy coef = 1.0, proxy\_lr = 3e-3**

****

**Accuracy = 0.7645**

1. **Triplet Margin Loss + Proxy Loss, margin = 0.2, semi-hard семплирование, proxy coef = 0.5, proxy\_lr = 1e-3**

****

**Accuracy = 0.8254**

1. **Faiss + KNN -** сделали замер для 4-х разных k. В качестве модели-эмбеддера использовали модель из 10-го запуска, так как она показала наивысшую метрику. Вот, какие получились результаты:
   1. **FAISS top 5 accuracy: 83.34%**
   2. **FAISS top 3 accuracy: 83.00%**
   3. **FAISS top 10 accuracy: 83.19%**
   4. **FAISS top 1 accuracy: 82.83%**

# Выводы

* batch-hard семплирование оказалось хуже, чем semi-hard. График loss-а слишком расшатанный, loss начинает вести себя непредсказуемо.
* Во всех экспериментах мы замечали, что крайне важно подбирать гиперпараметры. Так, margin в 0.2 дал прирост к качеству, а в случае proxy loss-ом нужна была тонкая настройка.
* Proxy Loss сделал обучение стабильным и плавным - это видно по графику - а также повысил итоговую метрику.
* Более умное семплирование также накинуло 1% к качеству.