Система распознавания музыкальной транскрипции при помощи методов машинного обучения

Магистрант: Андрадэ А.И.

Руководитель: Насуро Е.В.

Цели и задачи

- Предложить свой метод для решения задачи транскрибирования фортепиано
- Реализовать его
- Поработать над улучшением результата, внося изменения в модель
- Если результат перестает улучшаться, предложить новые пути по достижению лучшего результата

Гипотезы

- Частотно-временное представление произведения фортепиано содержит достаточно информации для выделения признаков, по которым можно точно определить наличие или отсутствие каждой ноты в определенный момент времени.
- Временная составляющая свойств произведения может быть хорошо обработана за счет архитектурных особенностей рекуррентных сетей типа LSTM (Long-Short Term Memory) без введения отдельных шагов алгоритма.
- Возможно обучить единственную нейросеть, которая будет выполнять бинарную классификацию для каждой из 88 нот независимо.
- Нейросеть самостоятельно найдет и выделит лучшие свойства для классификации из частотно-временого представления (глубокое обучение)

Датасет

- Подходит ли датасет для решения задачи ?
  - *http://www.piano-midi.de*
  - 25 композиторов
- Достаточно ли в нем данных ?
  - Взято 305 произведений для обучения
  - Длинна произведения от 1 мин до 12 мин.
- данные представлены в удобной форме ?
  - Данные в форматах mp3, ogg для обучения на признаках
  - Midi для меток

Подготовка данных

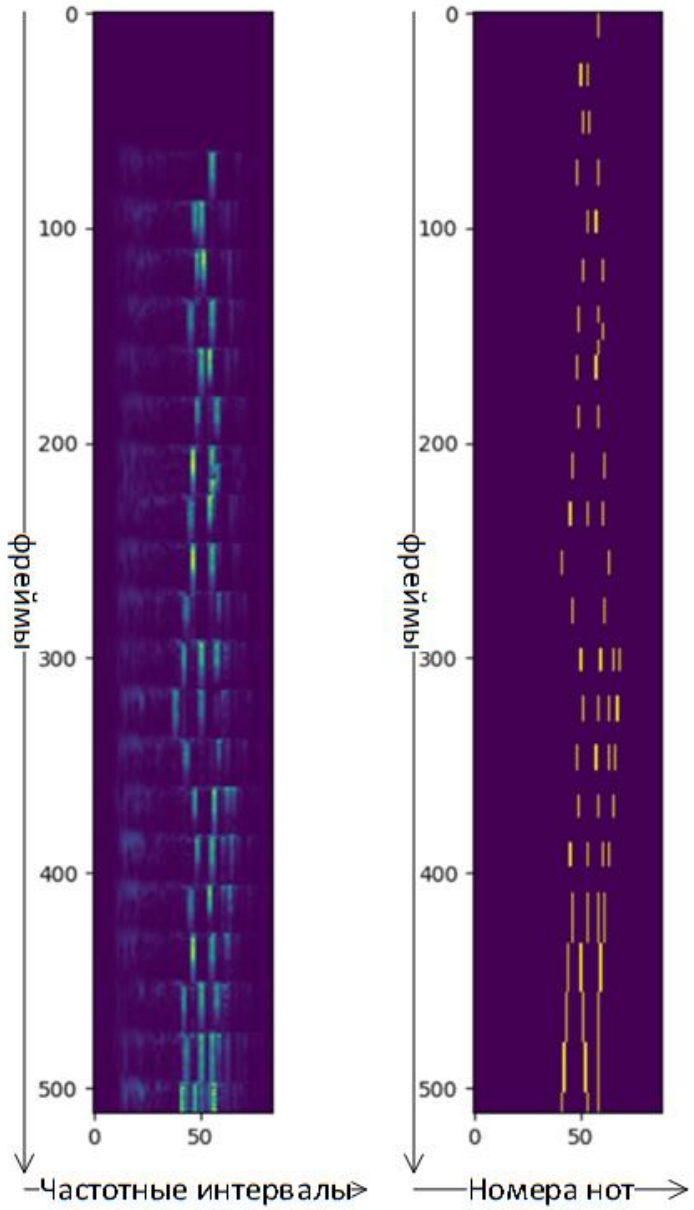
- Какое временно-частотное представление выбрать ?
  - Преобразование Фурье (невозможно одновременно обеспечить хорошее разрешение по времени и по частоте, разрешение по осям является постоянно)
  - Constant-Q (последовательность логарифмически разнесенных фильтров)
  - Непрерывное вейвлет преобразование (слишком вычислительно дорогое)
- Какая библиотека python работает с audio ?
  - Librosa
- Какая библиотека python работает с midi ?
  - python-midi (как оказалось, библиотека не умеет переводить midi ticks в секунды)
  - pretty_midi

Данные

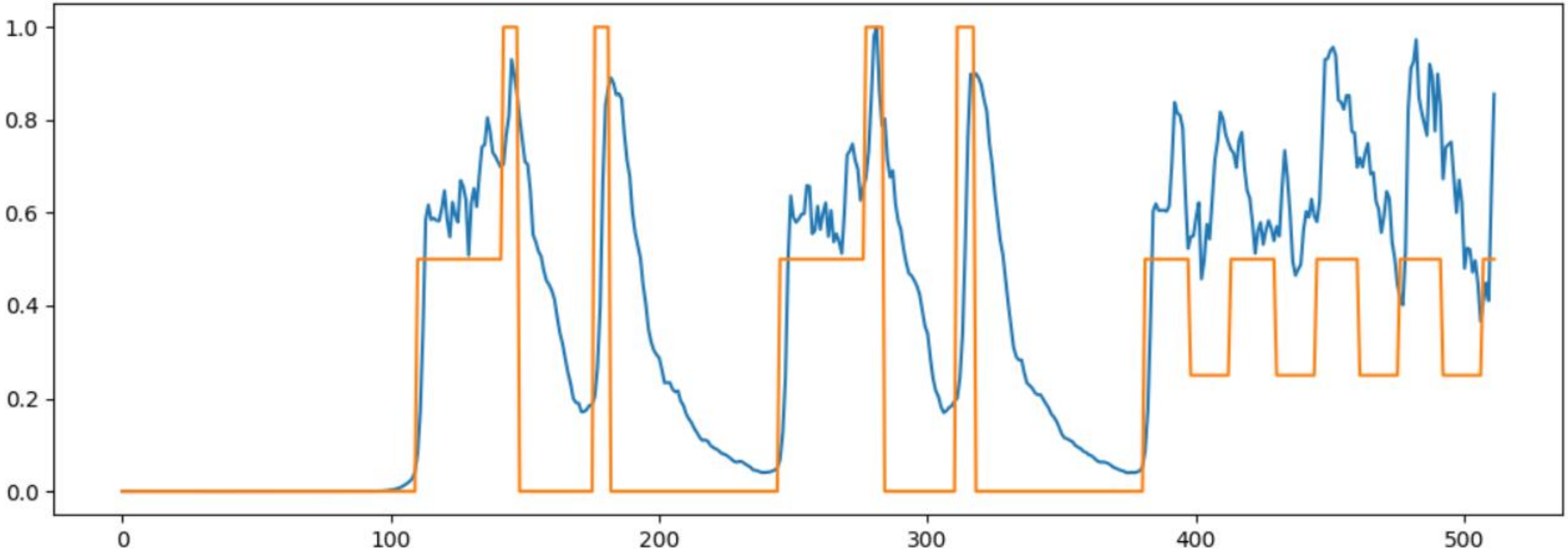
На рисунке видно смещение свойств относительно midi-меток.

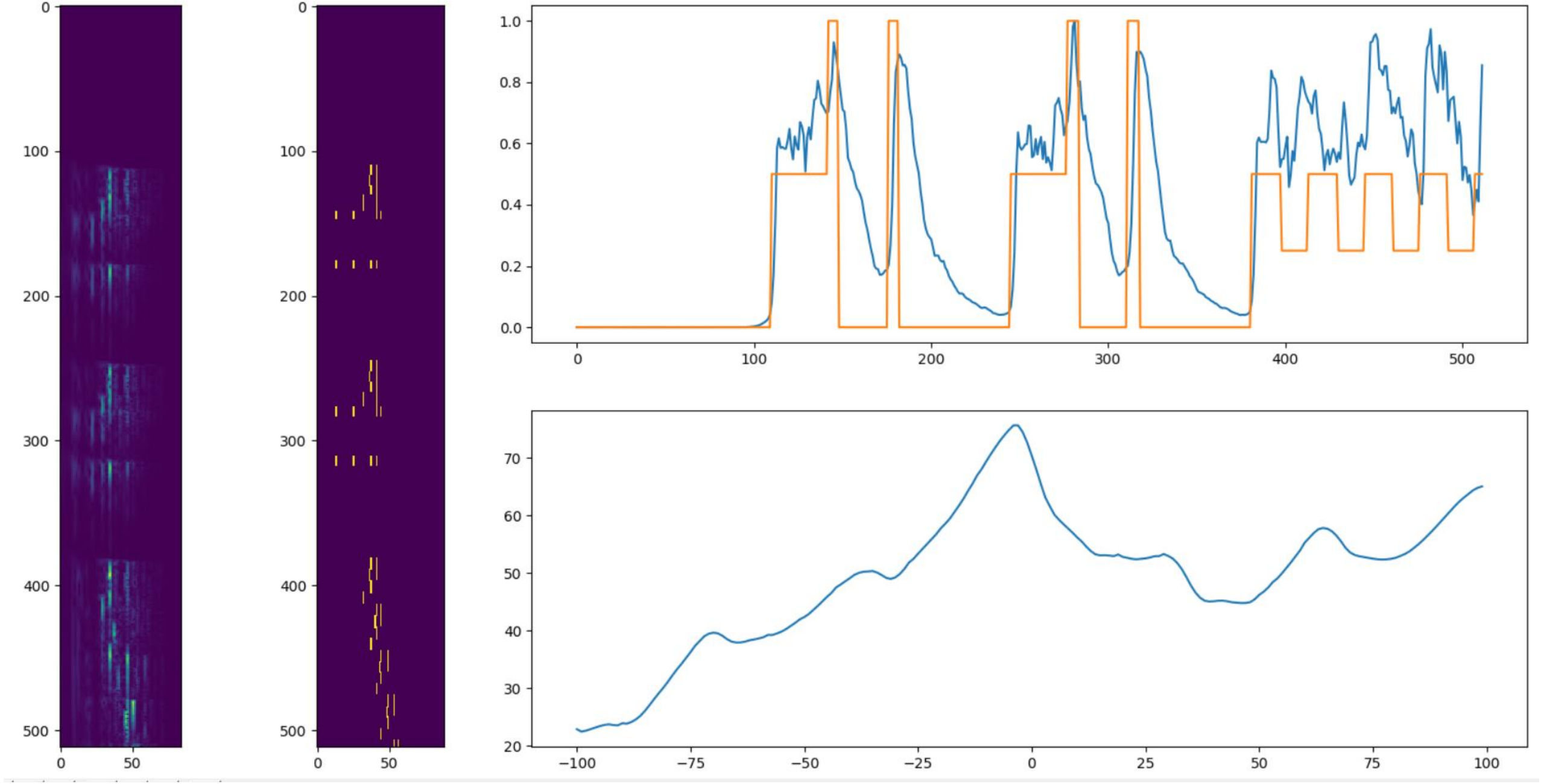
Как точно выровнять тренировочные данные по времени (фреймам) ?

Ответ: корелляция, притом одномерная. Значит нужно привести данные к одномерному виду, например суммировав значения амплитуд всех нот для каждого фрейма



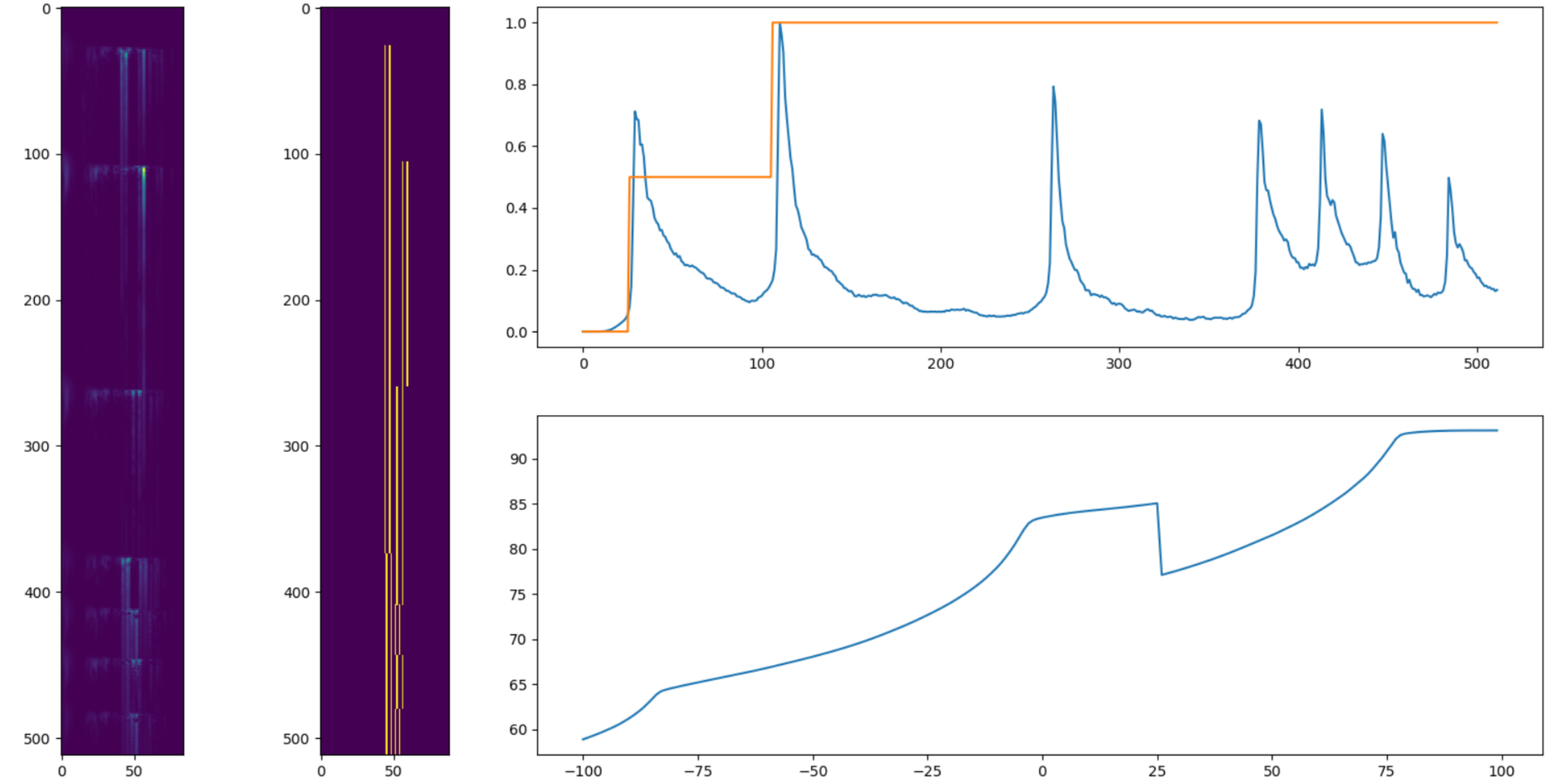
Корреляция




0
100
200
300
400
500
0
50
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
70
60
50
40
30
20
−100
−75
−50
−25
0
25
50
75
100

Корреляция

- Можно ли узнать относительное смещение для всех произведений с помощью корреляции ?
  - Как выяснилось, что не так просто


0
100
200
300
400
500
0
50
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
90
85
80
75
70
65
60
−100
−75
−50
−25
0
25
50
75
100

Данные

- Все произведения имеют разную длину, соответственно кол-во фреймов. Какой формы должны быть данные для обучения ?
  - Что если разбить все произведения на участки одинаковой длинны, например по 512 фреймов ?
  - Размер участка должен быть таким, чтобы вмещать по продолжительности хотя бы несколько нот, так как сеть должна обучится работать с временной составляющей данных
  - Сеть запоминает временные зависимости в пределах одного участка

Генератор данных

- Объем данных велик, в numpy формате – 8 ГБ, как обучать, если привык загружать данные в оперативную память ?
  - Использовать генератор, для этого надо наследоваться от keras.utils.Sequence базового класса и реализовать методы __len__ и __getitem__.

Гиперпараметры

- момент и скорость обучения
- количество скрытых слоев
- количество нейронов в каждом слое
- процент dropout
- кол-во эпох обучения

Гиперпараметры

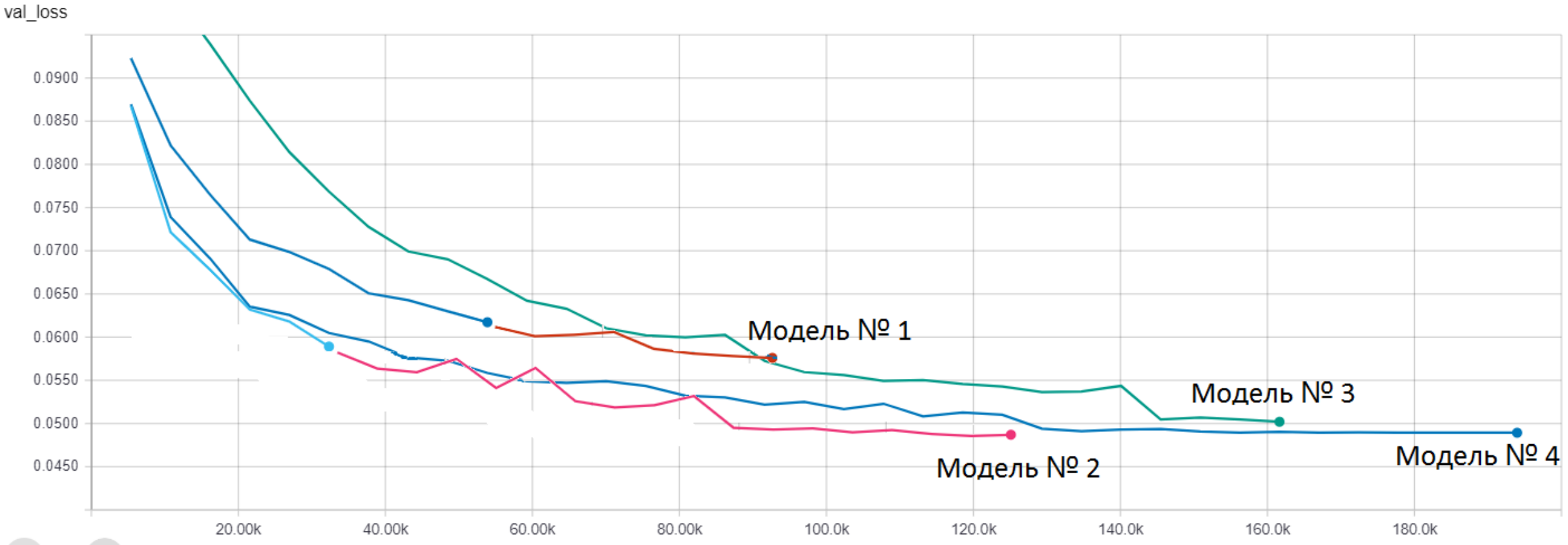
- Можно ли как-то рассчитать гиперпараметры ?
  - **Нет**, можно слегка лишь прикинуть, в основном они определяются эмпирическим путем
  - Например, прикинуть кол-во нейронов в скрытом слое можно так:

$$N_h = \frac{2}{3} \cdot (N_i + N_o) = \frac{2}{3} \cdot (84 + 88) \approx 115$$

Модель данных

```
model = Sequential()
model.add(LSTM(120,
               dropout=0.2,
               recurrent_dropout=0.2,
               input_shape=(512, 84),
               return_sequences=True))
 model.add(TimeDistributed(Dense(120, activation='relu')))
 model.add(TimeDistributed(Dense(88, activation='sigmoid')))
 model.compile(loss='binary_crossentropy', optimizer='adam')
```

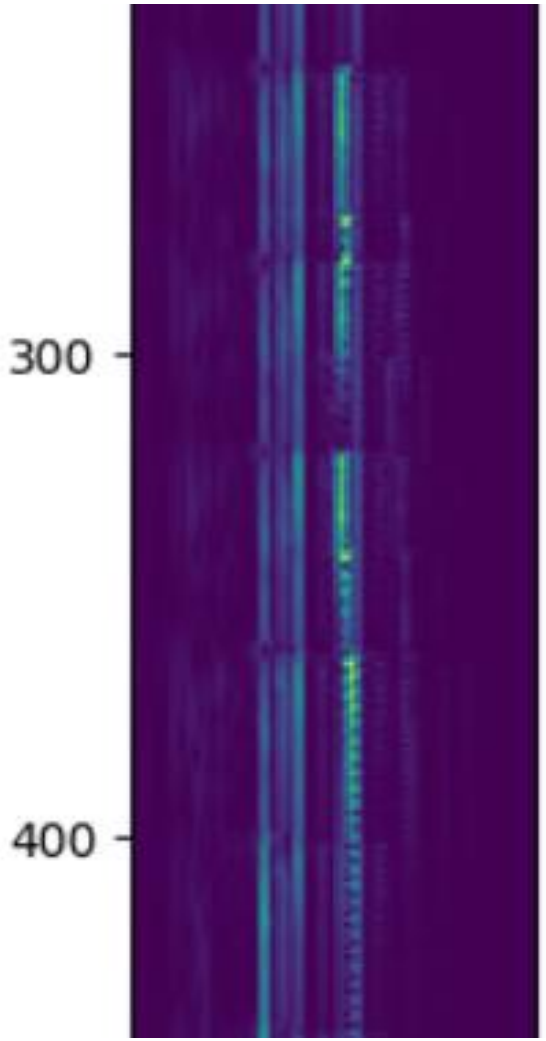
Обучение

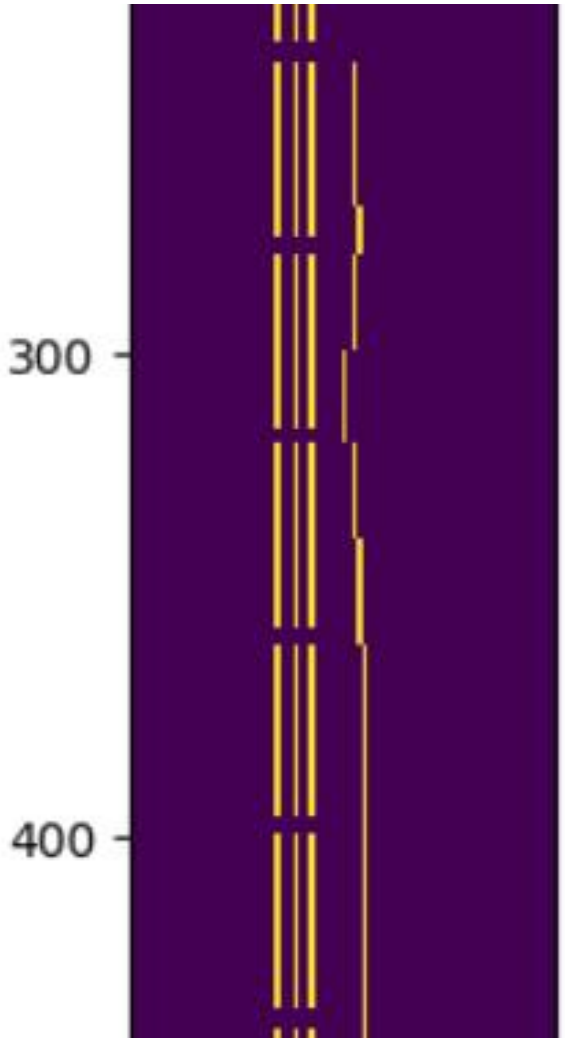

val_loss
0.0900
0.0850
0.0800
0.0750
0.0700
0.0650
0.0600
0.0550
0.0500
0.0450
20.00k
40.00k
60.00k
80.00k
100.0k
120.0k
140.0k
160.0k
180.0k
Модель № 1
Модель № 2
Модель № 3
Модель № 4

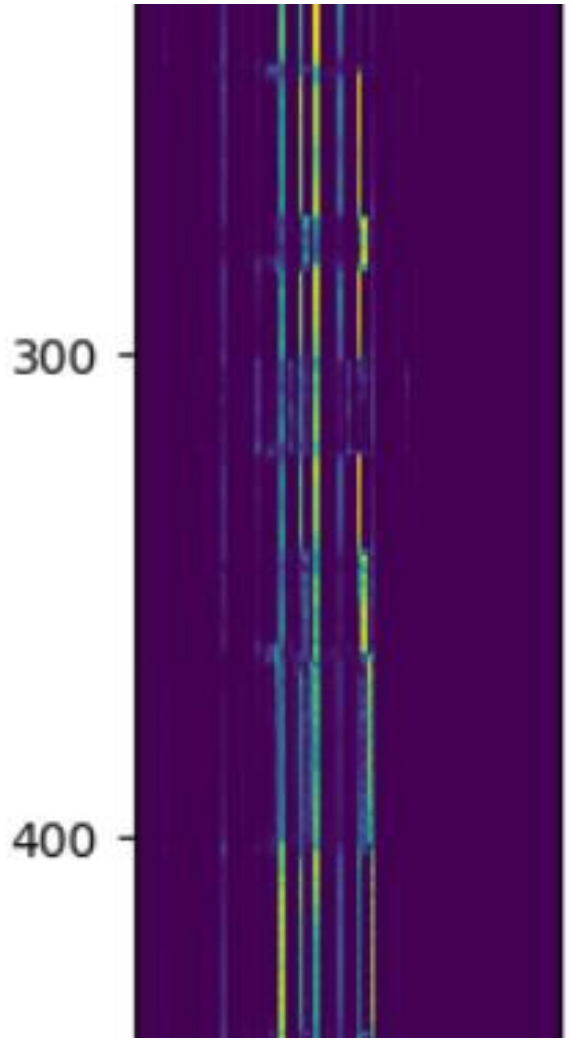
Обучение

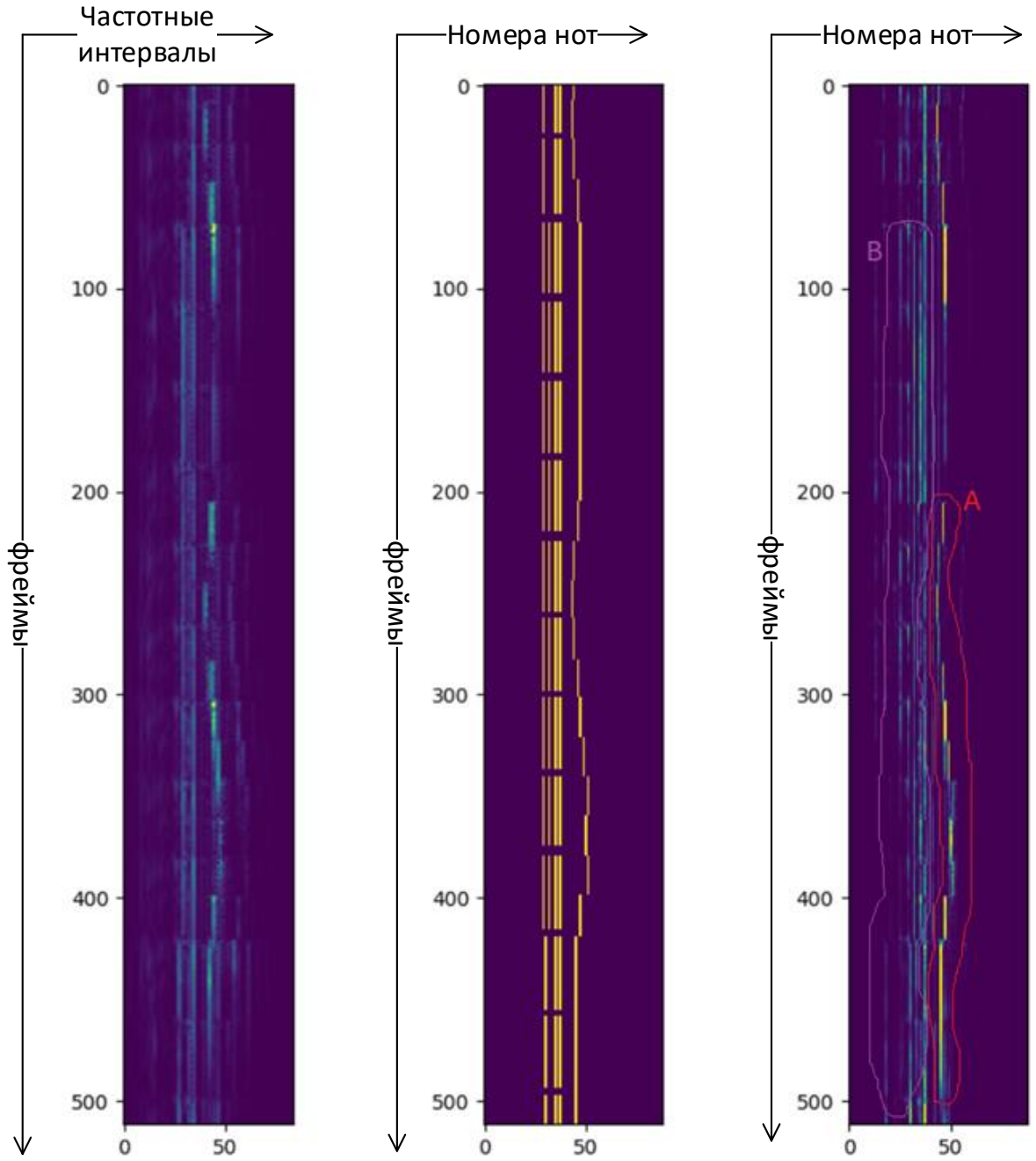
- Кол-во эпох разное, от 10 до 100
- Время обучения одной модели на Nvidia Cuda ~ сутки
- Используются функции обратного вызова
  - для сохранения модели вместе с весами в конце каждой эпохи
  - Для прекращения обучения, если функция потерь вернет NaN
  - Для уменьшения скорости обучения, если значения функции потерь не уменьшается более 2 эпох подряд

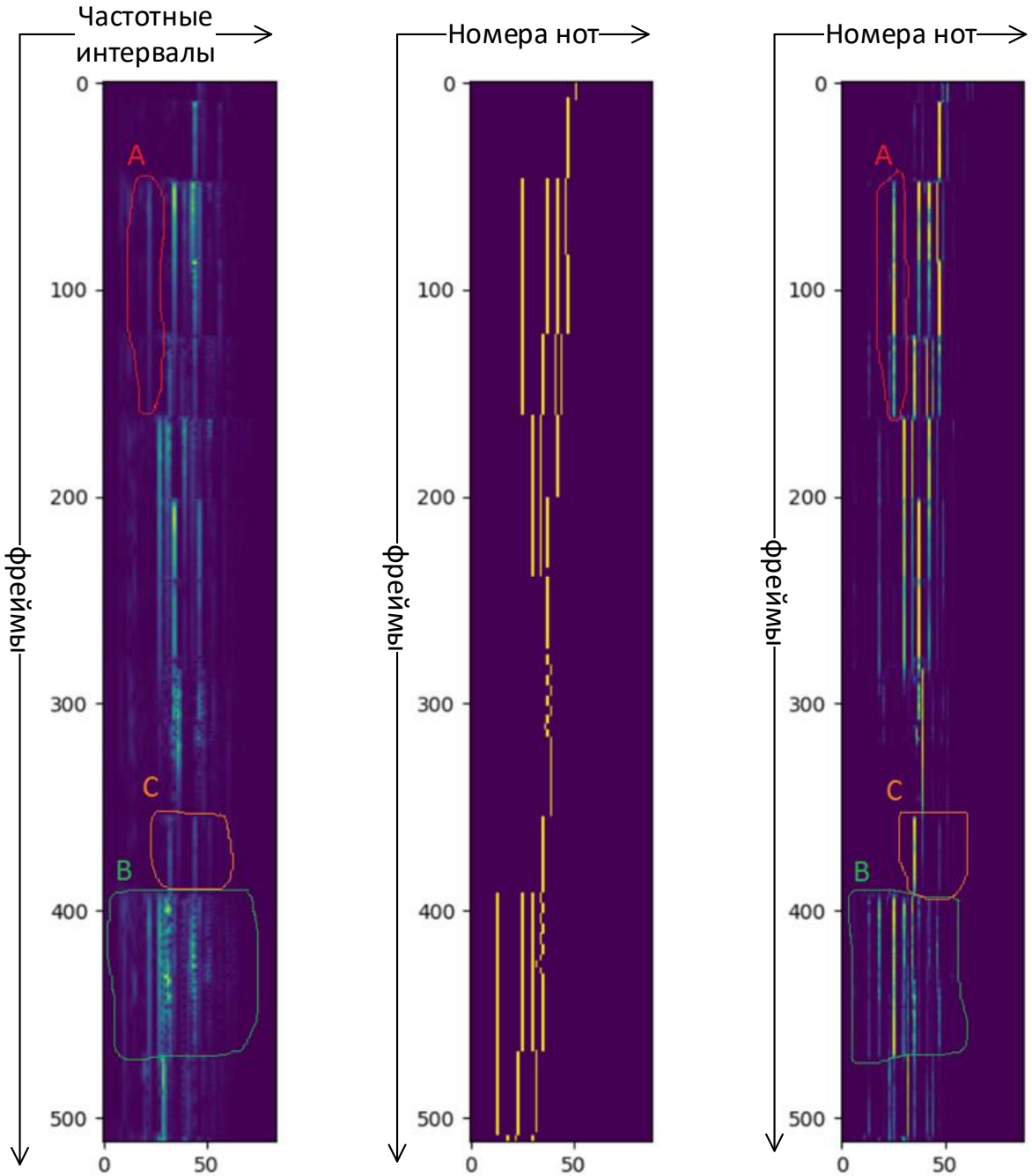
Результаты обучения

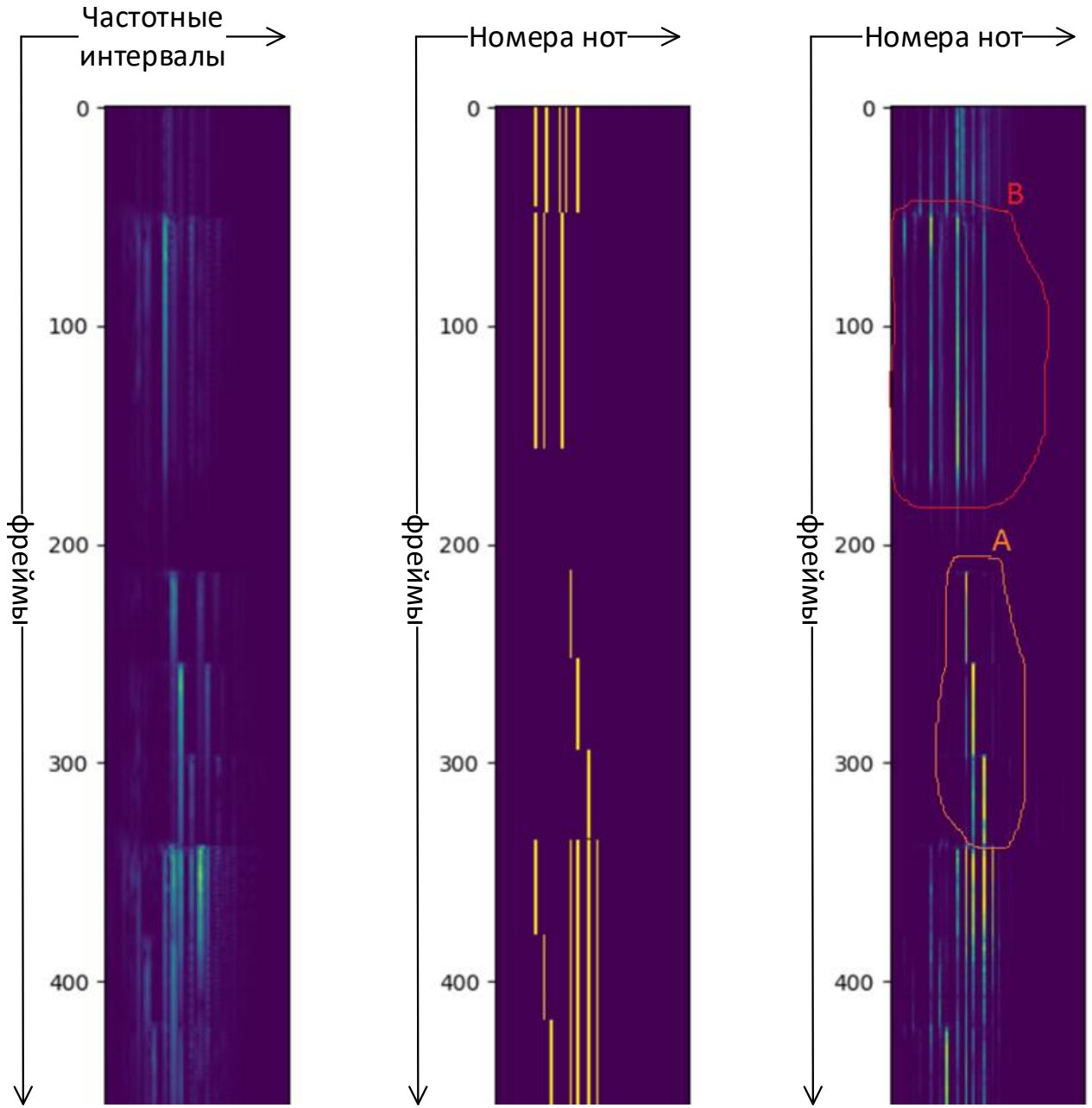


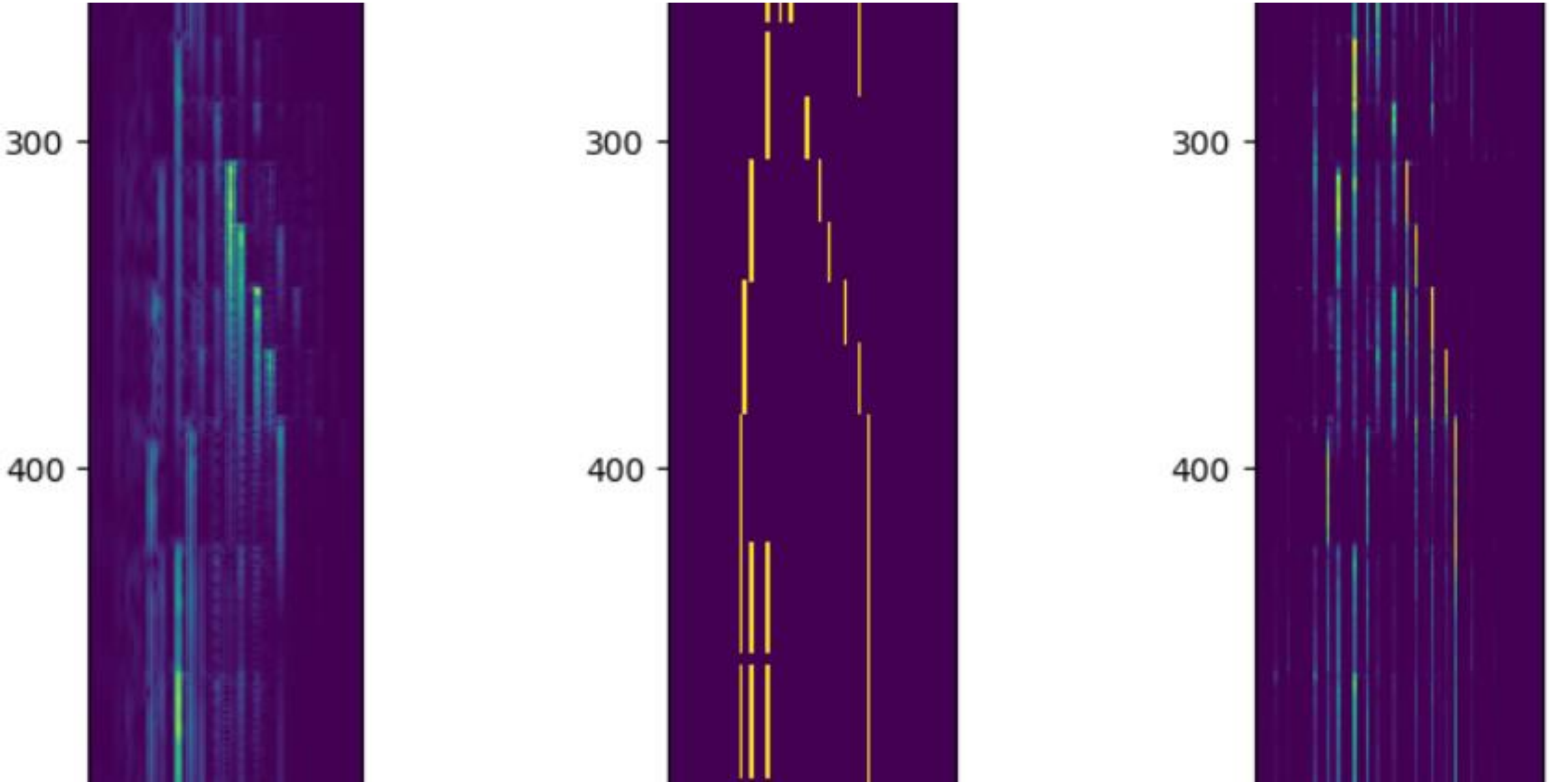





Частотные интервалы
фреймы
0
100
200
300
400
500
0
50
Номера нот
фреймы
0
100
200
300
400
500
0
50
Номера нот
фреймы
B
A
0
100
200
300
400
500
0
50


Частотные интервалы
Номера нот
Номера нот
фреймы
фреймы
фреймы
A
B
C
0
100
200
300
400
500
0
50


Частотные интервалы
Номера нот
Номера нот
фреймы
фреймы
фреймы
0
100
200
300
400
B
A


300
400
300
400
300
400

Как улучшить ?

- Больший датасет ?
- Дополнительно обучить звучанию отдельных нот ?
- Увеличить кол-во частотных интервалов (bins) ?
- В 2018 г для соревнования на kaggle был выложен в доступ новый датасет с 200 часов фортепианных выступлений, записанных с точным выравниванием (~ 3 мс)
- Другая архитектура сети ?

Спасибо за внимание !