Regarding the background knowledge for onset detection, why do you think that this is a challenging problem?

To detect the onset, especially the **string instrument** music onset is difficult than normal instruments such as flute or piano, which the curve is steep. The frequency level between two notes of string music changes gently. And also with several events such as **ornamentations** or **glissandos** happened in music, the normal decisively predict often have many wrong predictions.

To solve such problems, we need to set an allowable range that the frequency (or other features that CNN learned) change within this range are not wrongly predicted as onset. By setting this "tolerance", the processing, training and evaluation method need to change a lot. We also need to develop our own method for evaluation and find the best model. (For example, give about 2 frame tolerance for evaluation.

Please can you clarify the software you produced for your keras-based CNN system? Which part did you spend the most time on?

```
import keras.backend as K
import sklearn
import tensorflow as tf
from tensorflow import keras

#split the data

X_train, X_test, Y_train, Y_test = \ sklearn.model_selection.train_test_split(img_data, label_data, test_size=0.2)

#Create the model
bench, height, width, channels = X_train.shape

model = keras.models.Sequential([
    keras.layers.Conv2D(filters=10, kernel_size=(3, 7), strides=(1, 1), padding="VALID",
activation="tanh",input_shape=[height, width, channels]),
    keras.layers.MaxPooling2D(pool_size=(3, 1), strides=None, padding="VALID"),
    keras.layers.Conv2D(filters=20, kernel_size=(3, 3), strides=(1, 1), padding="VALID", activation="tanh"),
    keras.layers.MaxPooling2D(pool_size=(3, 1), strides=None, padding="VALID"),
    keras.layers.Flatten(),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(256, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1, activation="sigmoid")
])
#Compile the Model
model.compile(loss=" binary_crossentropy ",
              metrics=['accuracy'],
              optimizer= \ keras.optimizers.SGD(learning_rate=0.02,momentum=0.45))  # 0.05 0.45
#Train the Model
history = model.fit(X_train, Y_train,
                    batch_size=256, epochs=300, verbose=1,\ validation_data=(X_test, Y_test))
#Evaluation the model
scores = model.evaluate(X, Y, verbose=0)
#Predict the result
predictions = model.predict(slice_for_predict)
#save the model
model.save('model.h5')
#load the model
keras.models.load_model('model.h5')
```

Above is the whole software coding in Pycharm with the steps of:

1. Split the data
2. Create the model
3. Compile
4. Train
5. Predict
6. Save

Actually I spent a lot of time on modifying the model structure(activation function, dropout), change loss function, and justify the optimizer and it's learning_rate and momentum. For activation function in max-pooling layer and fully connected layer, I tried many conbinations such as "relu" follows "sigmoid", "tanh" with "sigmoid" and so on. Change optimizer to Adam or others. Also tuned the learning_rate in range of 0.05 to 0.001, momentum changes from 0.9 down to 0.2. Different combinations performed totally different and overfitting and underfitting also happened many times. For each combination of above I'll save the model and check the accuracy curve. Finally I'll load these models and compare their prediction ability and try to find the better training way and it's structure.

With regards to other techniques not using neural networks for onset detection, in which contexts do you think these would be preferable?

For a very simple method : just use the next note's power spectrum to minus the previous one, we simply looking the local maximum in the mean amplitude change. These traditional signal processing methods fits more on small set of data, or those who can't get enough labelled data for training(normally need more than 10000 for good performance). Also in other music analysis area such as pitch detection and music transient analysis, they need to focus more on their own topic rather than specifically training another onset model, they can simply calculate the difference in frequency and find the beginning of signal. In some online software for music recognition, they actually do this.

Please justify your choice of Convolutional Neural Networks. While these are current state-of-art, why do you think they have yet to achieve 100% accuracy?

The predict accuracy determined by model trained well or not. And the model training is highly relied on training data.The training data can't cover all the possible conditions (Large data set can only make probability close to 1 but never achieve). Also the 100% accuracy in training always means over trained rather than well predicted.