Deep Learning Final Project

Daniele Acquaviva (i6229921)

May 2020

1 Introduction

In the following project, different models to correctly classify brain data are explored, namely Intra and Cross classification is performed. Different metrics are used to assess the models' performances during the phase of hyperparameter tuning in order to select the best model. Data preprocessing is implemented, to scale and downsample the data. Finally, alternative approaches are considered to further increase the model's performances.

2 Data description

In the following project brain data are used, namely MEG brain data. Each subject corresponds to a file, named with the following format, "taskType, subjectIdenfier, number". Task type represents the label of the subject, indicating that the data were collecting while doing a particular activity, namely the considered classes are Resting task, Math & Story Task, Working Memory task. SubjectIdenfier, indicates a particular subject, whereas number represents a different measurement of a particular subject. Each file contains data collected with a sample rate of 2034 Hz, therefore every file corresponds to a duration of 17.5 seconds, resulting in a 35624 time step signal per channel. Multiple sensors (magnetometers) have been used to collect those data, each sensor is considered to be a channel, the data contain, 248 channels. As first decision to be made, each file is considered to be a single sample. Therefore, goal of this project is to classify correctly each file, namely 17.5 seconds of data.

3 Memory management during training

Since during the workflow there might have been some memory issues, due to the size of the data, a data generator is implemented and provided alongside this report. For intra classification it was not needed since the training sample are only 32 and the model used to classify those data is less computational demanding. However for one of the model implemented for cross the data generator is used, namely with the multi head module. The data generator module load one file at the time from the disk identifying each sample by their file name. Therefore each sample needs to be named with an id. The data generator can be found in one of the file provided along side this report, namely "*datagenerator.py*"

4 Preprocessing data

The order of magnitude of the data is $10e^{-15}$, therefore they might not be adapted for deep learning tasks. To overcome this, different preprocessing data techniques are implemented. They are all included in one of the file provided along side this report, namely, "*preprocess.py*".

4.1 Min-max scaling

Each channel is tread independently, therefore a min-max scaling is applied to each channel over all training samples. This is done to preserve the distribution of data on each channel, each magnetometers is placed on a different position of the human scalp, therefore they may capture different features. The same min and max values of the training set are also used

to scale the test set before feeding it to the network to perform classification. This is done, to ensure the robustness of the model that is trained on the training set, therefore, might only handle values in the range of the training set as well as to keep consistency during the workflow.

4.2 Downsampling

Downsampling is also applied to reduce the dimensions of the data along the time step dimension. This is done, by considering that not every measurements in the time series is significant, therefore the frequence at which the data are stored is reduced by a sampling rate factor. The sampling rate simply indicates that during the downsampling, one element is kept every k elements, where k is equal to the sampling rate. A further justification of downsampling is that it makes deep learning training faster, while not necessarily having a negative impact on the accuracy. Since the choice of different sampling rates may lead to completely different results, it is considered to be an hyperparameter of the model, therefore is tuned in order to select the best one. Moreover, it is good to point out that, downsampling is performed before scaling the data, in this way, we ensure that the min value and max value considered to perform min-max scaling are not deleted during downsampling.

4.3 Outliers detection

To make sure that the observations best represent the problem, outliers detection is performed. Sometimes a dataset can contain extreme values that are outside the range of what is expected and unlike the other data. In this project an outlier is considered to be a value that is five standard deviations or more from the mean of the values in a single channel. If an outlier is detected, all values present in the channel are replaced with the mean of the elements of the channel.

5 Evaluation

Both for Intra and Cross, different metrics are used to asses models' performaces. Only the training folder is used for training purposes, whereas the test folder is used only as final evaluation, therefore no tuning or training is done based on the test folder, the results on the test folder are discussed in Section 8. The metrics used for both tasks are:

- Categorical Cross-entropy: It is used as loss function. The choice of using the cross-entropy error function instead of the MSE or others for a classification problem is driven by the fact that Cross-entropy leads to faster training as well as a good generalization.
- Accuracy: It is used to see how well the model is able to classify samples.

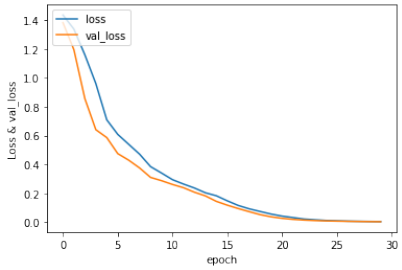
5.1 Intra subject classification

Cross-validation is used as a resampling procedure to iterately split, each time in a different way, the training set into two sub set, namely, training and validation. In particular Stratified-KFold validation is used, to ensure that the validation set contains an equal distribution of samples belonging to a particular class, this is done for every fold and each time the validation set contains different samples. This procedure, aims to keep the model's performances robust as well as to avoid overfitting, in other words the model should be able to generalized when used with new unseen data. Since the training samples present for intra are only 32, 5 fold are considered, in this way the number of training examples per fold are not too low, therefore the model is able to train and validate the performance on the validation set well. Each fold returns the loss (Cross-entropy) as well as the accuracy of the model, performed on the validation set, therefore the mean of these two metrics is return, indicating the general performances of the model in different runs (5 folds in this case). Those two values are then used to compare models and select the best one.

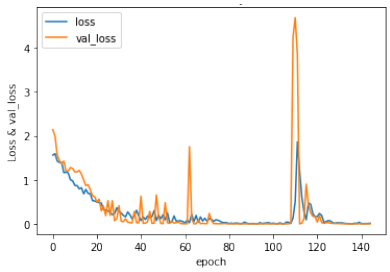
5.2 Cross subject classification

Cross subject classification uses training data from different subjects, therefore the training might result unstable with unexpected spikes of the loss as well as the accuracy, due to a completely different sample deriving from a different subject. In Figure 1 the training of Intra and Cross is shown, plotting loss and accuracy. It is possible to notice different spikes

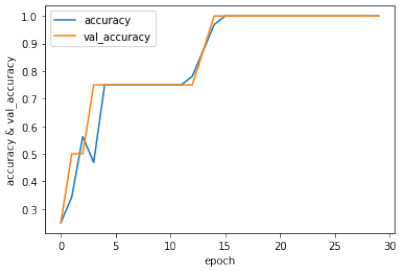
during the cross subject training, happening at the same epoch, for both loss and accuracy. This might indicate that, one of the batches considered at that epoch might contain very different samples. Due to this unexpected behavior, normal train-validation split is preferred over a Stratified-KFold validation for cross subject classification. In this way at the end of the training it is possible to plot and see how the loss validation and the accuracy validation are changing over time. Since only 64 samples are provided, only 8 samples are used as validation set, containing 2 samples per each class.



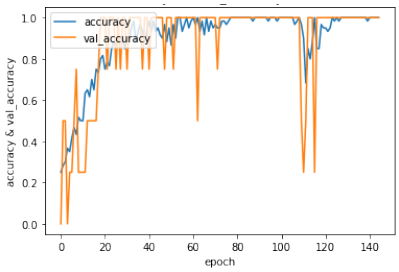
(a) Loss per epoch of train and validation set using Intra data



(b) Loss per epoch of train and validation set using Cross data



(c) Accuracy per epoch of train and validation set using Intra data



(d) Accuracy per epoch of train and validation set using Intra data

Figure 1: Training on intra and cross data, using a one of the model mentioned in section 6, intra presents a nice and smooth loss and validation whereas cross presents unexpected spikes.

6 Model

Since the data provided are sequences of data per channels, namely a multivariate time series, the core layers of every model implemented, is a 1D convolutianal layer. Different models are implemented with the aim of increase accuracy only for the cross classification, since intra classification resulted to be direct and straightforward after hyperparameters tuning. The models are listed here, and they increase with complexity (Pictures of the models architectures are included in the appendix of the report):

1. Baseline model: The baseline model is composed by two 1D convolutianal layers followed by two dense layers, with a flatten layer in between to flatten the output deriving from the conv layers into an array so that it can be passed to the next dense layer. Dropout layers are also explored to make the model more robust.

2. Baseline skip-connection model: Since adding more layers to the model did not improve the performances of the model but was actually decreasing it, a residual block, implementing a skip connections is used in the dense part of the network. This is done with the hope of increasing performances while increasing the model depth, making sure that the new and more deep model is good at least as the previous. Therefore, the resulting residual block allows the model to theoretically do not worse than the model without it, simply by learning the identity mapping of the input to the residual block. This is done by adding the output of a layer to the output of another layer, that is present later in the network. The addition of these to outputs happen right before the activation function of the second output.

3. Multi-head model: The model is composed by multiple convolutional layers with differently sized kernels to interpret the same input and extract different features. It is composed by two block, namely heads. The two heads are considered to be equal to the convolutional architecture part of the baseline model, therefore two conv layer. Finally, the output of the two heads are concatenated and fed into a dense layer connected to another dense and the output layer.

4. Multiple Input Model: The model is composed by two heads, that are again equal to the baseline model, however the difference here, is that the model takes two versions of the input. The first input only consider the first 124 channels of each sample and is passed to the first head, whereas the second input, consider the remaining 124 channels of each sample. In this way each head can separately extract features from different inputs and then merge them together to produce a final output.

5. 1D_CNN-Autoencoder: A 1D_CNN-Autoencoder is implemented with the aim of performing features extraction, therefore reducing the dimension of the original input, due to the high dimension of the timeseries of each channel. Once the autoencoder is trained the decoder part is removed and replaced with the baseline model to further extract feature and perform classification in the final layer. A deep CNN should already be able to extract features and therefore perform feature extraction, unfortunately this did not happen for the cross subject classification task. The autoencoder is trained trying to reconstruct the original input in the output layer.

7 Random Search

Due to time and hardware limitations, hyperparameters tuning is performed using the Baseline model for intra classification as well as cross classification, whereas the Multi-head module is tuned only for cross classification, for a total of 503 different models configurations and a total of 25 hours of computational time. Random search is used to best optimize and tune the models, where random combinations of the hyperparameters are selected. Random search does not require a list of combination of hyperparameters to follow, but instead, to specify ranges of values for every hyperparameters, therefore during the search, it was possible to stop and continue the search, whenever it was needed, starting again from a random configuration. Instead a grid search would have required to keep track of the hyperparameters already tuned, in order to start from the same point as well as having the constrain of finishing all possible combinations, this results in adding more complexity to a problem where already time and computational power are a problem.

7.1 Baseline Model Random Search

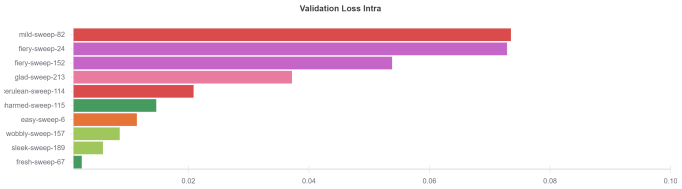
The ranges of the parameters configurations on which the random search is done for both intra and cross classification using the baseline model are summed up in the following table:

Hyperparameters	Min/Max - Categorical value Value
Batch_Normalization	True/False
Batch_Size	Min: 8 Max: 32
Number filters first conv layer	Min: 15 Max: 65
Number filters second conv layer	Min: 30 Max: 130
Dropout percentage layers	Min: 0.00 Max: 0.40
Number of Epochs	Min: 5 Max: 50
Kernel Size Conv Layers	Min: 2 Max: 15
Number of neurons first Dense layer	Min: 8 Max: 32
Number of neurons second Dense layer	Min: 15 Max: 65
Learning Rate	Min: 0.0004 Max: 0.0020
Optimazer	Adam/SGD
Sampling rate for downsampling	Min: 1 Max: 5
Weights Initialization	Zeros/Ones/Random_Uniform Random_Normal/Glorot_Uniform/Glorot_Norma

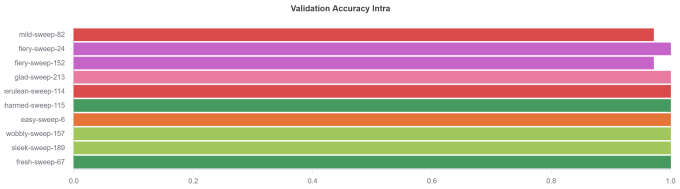
A visualization of the baseline model random search of intra as well as cross is present in the appendix.

7.1.1 Results compared for intra and cross using baseline model

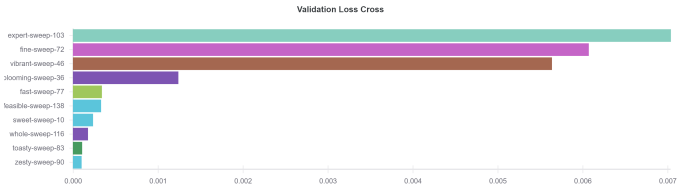
The baseline model is used and therefore tuned on both tasks. The hyperparameters selection during tuning are based on the validation metrics results, namely validation accuracy and validation loss are used. Since many configuration of hyperparameters result to have validation accuracy equal to 1 the validation loss is used to make a final judgment, choosing the model with the lowest validation loss.



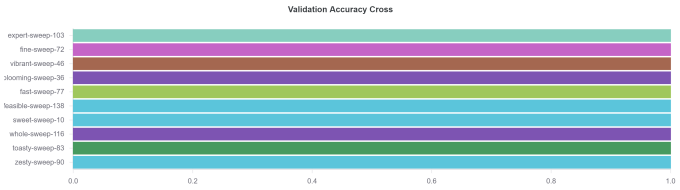
(a) Best Baseline Models validation loss for intra classification



(b) Best Baseline Models validation accuracy for intra classification



(c) Best Baseline Models validation loss for cross classification



(d) Best Baseline Models validation accuracy for cross classification

Figure 2

In figure 2 the best model configurations are shown after the random search for both intra and cross classification. Each model is named to best keep track of them (shown on the y-axis) and the relative validation loss and accuracy is shown as well. It is possible to notice that, all of them have validation accuracy equal to 1, therefore it is not a good metric to compare models, on the other hand the validation loss is different for all of them, therefore it can be considered to be a good metric to compare models. From figure 2 it is also possible to notice that the validation loss for the cross classification reaches lower values than the intra. As mentioned in section 5 a possible reason might be that while intra is evaluated using cross validation, cross is evaluated only using one validation set, therefore it is harder for models evaluated with intra to reach low losses since they are evaluated every time on different validation set. However, since the intra problem is direct and straightforward, they achieve impressive results anyway. The result for both type of task are quite impressive, since they achieve an accuracy of 1 and a very low loss, really close to 0, on the validation sets. Therefore selecting one of the model in figure 2 for both intra and cross should give good results on the test set, in other words they should be able to generalize quite well, with unseen data. The discussion on the test set is carried out in section 8.

7.1.2 Hyperparameter choice for intra classification

As mentioned already above a random search is performed to best select the hyperparameters for Intra classification. A visualization of the random search is present in the appendix. In figure 3 the relationship (correlation) between models' hyperparameters and validation loss is visualized.

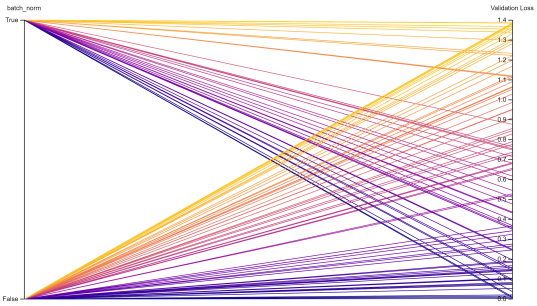
Figure 3: Correlation hyperparameters and validation loss for intra

Correlations capture linear relationships between individual hyperparameters and metric values, in this case the validation loss is chosen. Correlation values range from -1 to 1, where positive values (green bar) represent positive linear correlation,

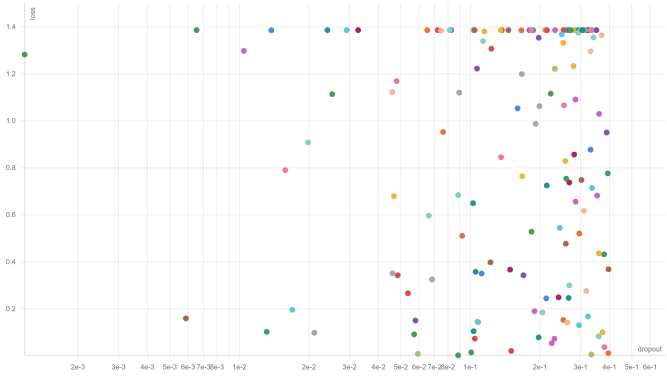
negative values (red bar) represent negative linear correlation and values of 0 represent no correlation. For a more accurate hyperparameter importance, the random search should have been run for longer, however from the figure above it is already possible to draw some conclusions, outliers are ignored to calculate the correlation between hyperparameters and validation loss, in this way a more reliable analysis can be done. Both numerical and categorical hyperparameter are shown. For weight initialization only 6 configuration are used, namely ones, indicating that all weights are initialized to 1, zeros, indicating that all weights are initialized with 0, random uniform, indicating that the weights are drawn from a standard uniform distribution, random normal, indicating that the weights are drawn from a normal standard distribution, glorot normal, indicating that the weights are drawn from a Xavier normal distribution and finally a glorot uniform, indicating that the weights are drawn from a Xavier uniform distribution. Moreover for optimizer only adam and stochastic gradient decent (sgd) are considered. Looking at the figure 3 we can say that, for numerical values, the higher the Batch Size, learning rate and Dropout the higher is the validation loss, since the bar is green, therefore we should prefer lower values of these hyperparameters. On the other hand a higher values of the remaining numerical hyperparameters indicates a lower validation loss, since the correlation is negative (red bar). Whereas for the categorical hyperparameters we have that most of the time when the hyperparameters are set with one of the following values the validation loss is high, due to a positive correlation. Namely, they are Optimazer sgd, Weight initialization using ones, Weight initialization using zeros and Batch normalization set to True. On the other hand, we can notice that most of the time when using the remaining categorical hyperparameter the validation loss tend to be lower since the correlation is negative (red bar). Although the numerical values are not present in the figure 3 it's very straightforward to understand the parameters importance, and therefore, prefer some to others.

7.1.3 Impact of batch normalization and dropout for intra

As required by the assignment the effect of batch normalization and the dropout are explored more in depth separately. Their impact is shown in Figure 4. Where in figure (a) the batch normalization is analysed, applied after each layers, whereas in figure (b) the dropout influence on the loss is shown, the dropout is added between the two conv layers as well as the two dense layers. In Figure (b) every point represents a different model configuration where on the x axis the dropout percentage is present and on the y axis the val loss is present. It is possible to notice that both of them do not have a significant impact over the validation loss, since for higher values of dropout the validation loss is both high and low, the same applies for batch normalization, when it is set to true and false the validation loss has high as well as low values. In Figure 3 their correlation is very low, reflecting the fact that both of them are independent from the validation loss. This conclusions might not reflect the true, therefore a longer random search might give different results, however the number of experiments performed in the random search is pretty high therefore we can consider this hypothesis to be reliable. Of course this conclusions only apply to the ranges of the hyperparameters and on the architecture of the model chosen.



(a) Impact of batch normalization on validation loss for intra

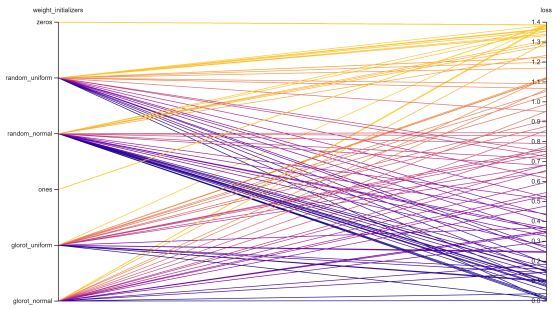


(b) Impact of dropout on validation loss for intra

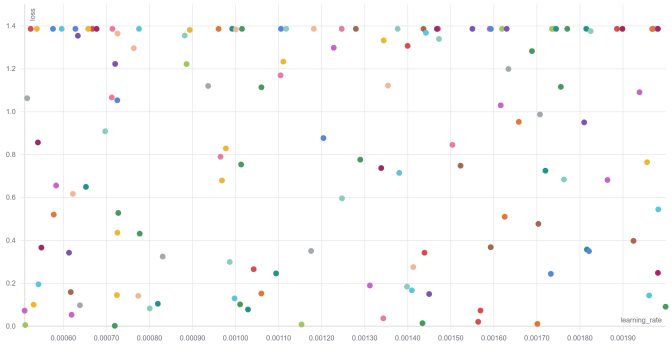
Figure 4

7.1.4 Impact of Learning rate and weight initialization for intra

As required by the assignment the effect of Learning rate and weight initialization are explored more in depth separately. Their impact is shown in Figure 5. It is possible to notice that the learning rate values do not have a significant impact



(a) Impact of weight initialization on validation loss for intra



(b) Impact of learning rate on validation loss for intra

Figure 5

over the validation loss, since the models configurations are spread over the entire plane, therefore for low and high values of learning rate there are high and low values of validation loss, therefore they can be considered independent, this is also reflected in figure 3. On the other hand the weights initialization do present a strong positive correlation for zeros and ones , whereas a strong negative correlation for the random normal weight initialization. Indeed, it is possible to see in figure (a) that from ones and zeros only yellow lines start, therefore they all end up to high value of loss, whereas for random normal, the lines leaving it are mostly blue, indicating they mostly end up to low values of the loss. For the others both yellow and blue lines are mostly the same. All of this is again shown in figure 3.

7.1.5 Hyperparameter choice for cross classification

A random search is performed to best select the hyperparameters for cross classification using the baseline model. A visualization of the random search is present in the appendix. In figure 6 the relationships between models' hyperparameters and validation loss is visualized, for the basile model on the cross classification task.

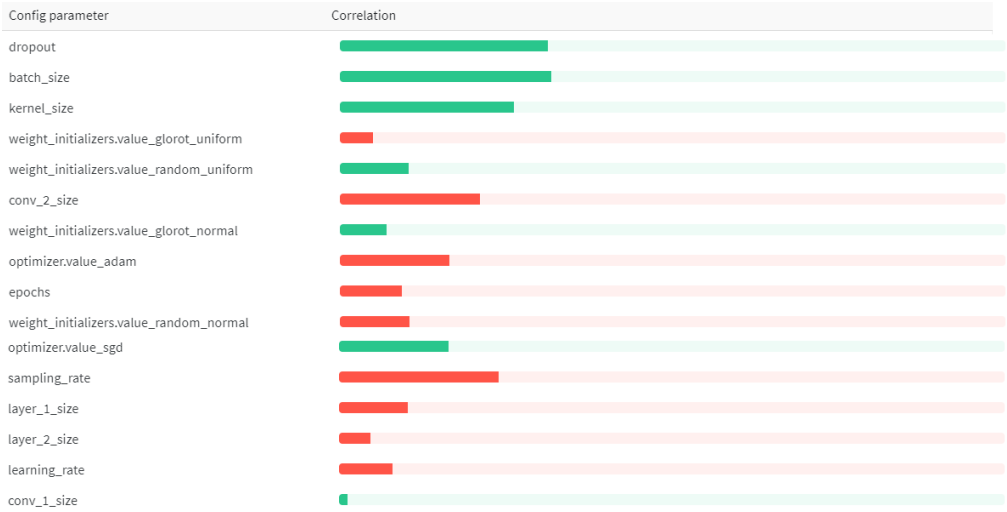
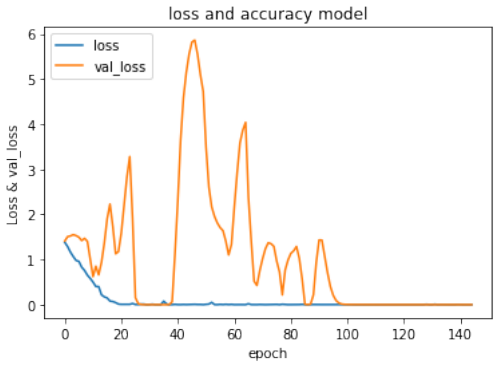


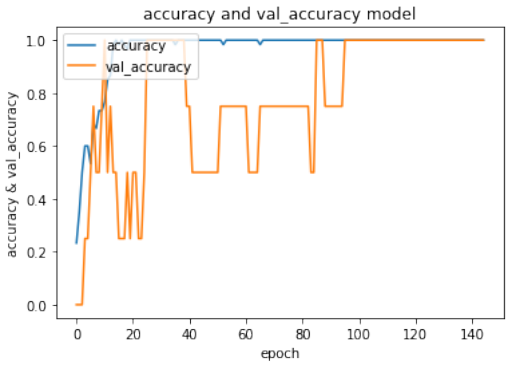
Figure 6: Correlation hyperparameters and validation loss for cross

Both numerical and categorical hyperparameter are shown. As specified in section 7.1.4, some weight initialization, namely zeros and ones, produced a very high loss, indeed they have a positive correlation with it. Therefore, they are excluded in the random search to speed up the process. Moreover the batch normalization is also removed from this random search, since it produces an unstable training, as shown in Figure 7.

Figure 6 indicates that dropout, batch size, kernel size and SGD have a high positive correlation with the validation loss, whereas the sampling rate, adam and the number of filters for the second convolutional layers have a high negative correlation.



(a) Impact of batch normalization on loss and validation loss for cross

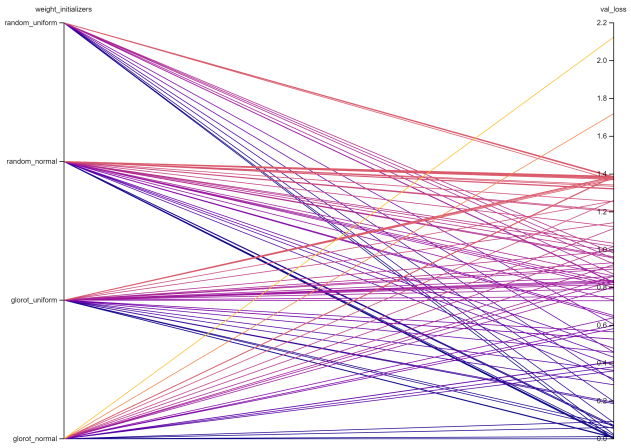


(b) Impact of batch normalization on accuracy and validation accuracy for cross

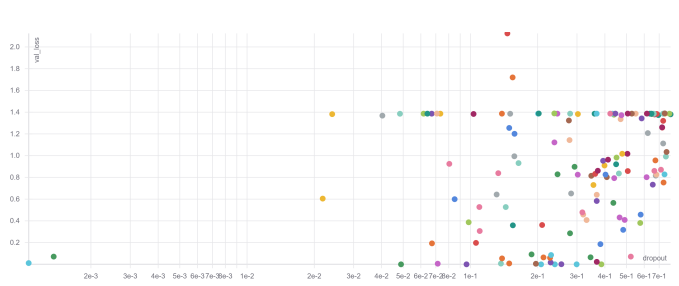
Figure 7

7.1.6 Impact of Learning rate, weight initialization and dropout for cross

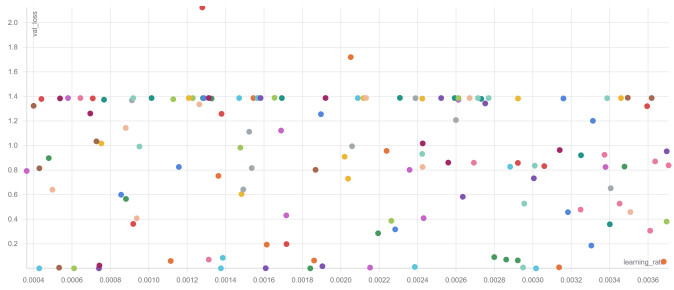
The impact of Learning rate weight initialization and dropout is shown in Figure 8. For learning rate and dropout it is possible to notice that both of them do not have a significant impact over the validation loss, since the models are distribuited over the whole plane. However since in the dropout figure (b) there are two outliers, having low values of dropout and very low loss, the correlation is drastically changed as shown in figure 6. Moreover the weight initialization does not seem to have any strong correlation with the validation loss, therefore any of them might be good to use.



(a) Impact of weight initialization on validation loss for cross



(b) Impact of dropout on validation loss for cross



(c) Impact of learning rate on validation loss for cross

Figure 8

7.2 Multi-head module Random Search

Since the Multi-head module seemed to be promising in term of low validation loss, random search is also performed for it. The ranges parameters configurations on which the random search is done are summed up in the following table:

A visualization of the Multi-head module Random Search of cross classification is present in the appendix. In figure 9 the

Hyperparameters	Min/Max - Categorical value Value
Batch_Size	Min: 5 Max: 65
Number filters first conv layer	Min: 10 Max: 130
Number filters second conv layer	Min: 30 Max: 130
Dropout percentage Conv layers	Min: 0 Max: 260
Dropout percentage Dense Layers	Min: 0.0 Max: 0.80
Number of Epochs	Min: 40 Max: 200
Number of neurons first Dense layer	Min: 50 Max: 100
Number of neurons second Dense layer	Min: 0 Max: 120
Learning Rate	Min: 0.0000 Max: 0.0040
Optimazer	Adam/SGD
Sampling rate for downsampling	Min: 1 Max: 100
Weights Initialization	Random_Uniform Random_Normal/Glorot_Uniform Glorot_Norma

best 10 models are shown in terms of validation loss, it is possible to see that the first model namely "$brisk - sweep - 77$" achieves a loss that is nearly zero, indicating a very robust and good model, able to generalize with new unseen data.

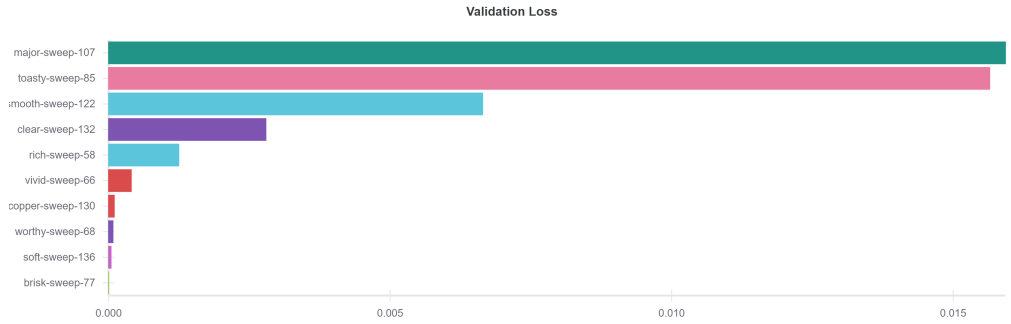


Figure 9: Best models random search multi head model

8 Results

After tuning, the model configurations selected are the ones for which the validation loss results to be the lowest for intra, whereas for cross, the ones for which the validation loss is the lowest and at the same time the training does not present any unexpected spikes of loss.
The best baseline model for intra is the first shown in Figure 2 (a) named "$fresh - sweep - 67$". The architecture is the following:

Hyperparameters	Value
Batch_Normalization	False
Batch_Size	32
Number filters first conv layer	46
Number filters second conv layer	117
Dropout percentage layers	0.08852
Number of Epochs	29
Kernel Size Conv Layers	6
Number of neurons first Dense layer	24
Number of neurons second Dense layer	43
Learning Rate	0.0007194
Optimazer	Adam
Sampling rate for downsampling	4
Weights Initialization	glorot_normal

Using the architecture shown in the above table, the accuracy achieved for the intra test is equal to 100% every time.
The best baseline model for cross is the first shown in Figure 2 (c) named "$zesty - sweep - 90$". The architecture is the following:

Hyperparameters	Value
Batch_Size	30
Number filters first conv layer	48
Number filters second conv layer	218
Dropout percentage layers	0.2072
Number of Epochs	122
Kernel Size Conv Layers	29
Number of neurons first Dense layer	45
Number of neurons second Dense layer	100
Learning Rate	0.003017
Optimazer	Adam
Sampling rate for downsampling	60
Weights Initialization	glorot_normal

Using the architecture shown in the above table, the best accuracy achieved for cross is: test 1 equal to 100%, test 2 equal to 50% and test 3 equal to 83%.
The results achieved using the best multi head module after tuning are very similar with the baseline model after tuning, therefore are not reported. Moreover the skip-connection model as well as the multi-input model achieve same performances of baseline model for cross classification therefore they are not reported either.
During cross classification test 1 and test 3 are most of the time well classified having accuracies equal to 100, 93 or 83 percent. Whereas for test 2 the accuracy achieved most of the time is 50%. However, sometimes the multi head module and the multi input module scored above 50% reaching 75% in test 2 and the training and validation loss can be seen in Figure 11. Figure 11 shows the training of the multi head model, it is possible to notice that there are no spikes and therefore the training is really stable. This might be the reason for achieving such result, however when scoring a values grater than 50% in test 2, the scores of test 1 and test 3 decrease. Since test 2 is the only subject for which the accuracy is hard to increase, a further investigation is conducted. In Figure 10, an heatmap is shown, representing the classifications of the model on test2. It is possible to notice that the model miss-classifies all samples of Math with working memory and all working memory tasks samples with math most of the time.

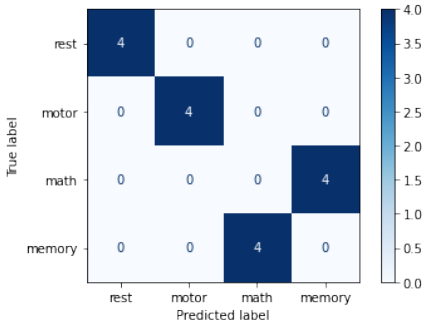


Figure 10: Heatmap showing classification for test 2

Different are the conclusions that can be drawn at this point.

1. The class math and memory are really similar, therefore is hard to capture the real pattern governing these two classes and therefore to make a distinction between them.

2. Test 2 data might contain some noise or outliers.

3. During labelling of the files or collection of the data an error has occur, therefore there might be a mismatch between the two classes.

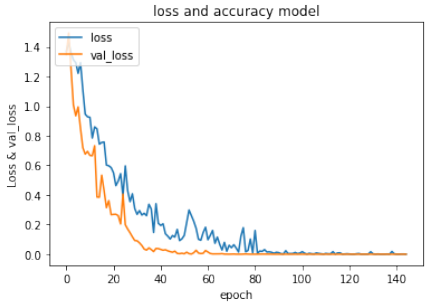


Figure 11: Training model when achieved 75% accuracy on test 2

9 Alternative approach

Different alternative approaches have been explored to increase further the performances on cross classification. As mentioned in section 6 different are the alternative solution proposed, starting from introducing skip connections to further increase the depth of the network without losing in performances, to different models with multiple inputs at the same time to try to capture and extract features at a different resolution and using different channels. Finally, since none of the model were able to increase the score on test 2, therefore not able to correctly extract the right features to perform classification from the original input , an autoencoder is implemented. Unfortunately the autoencoder training was really slow, making progress very slowly, therefore, it is not considered to be a functional alternative approach. Figure 14 shows the autoencoder training, it is possible to see that after 5000 epoches the loss decreased only from 0.96 to 0.63. A different alternative approach that might give good result is the use of an LSTM's in combination with the 1d CNN. LSTM's would better capture the temporal features of the signals present in each channel, therefore benefit more from the sequence structure of the data.

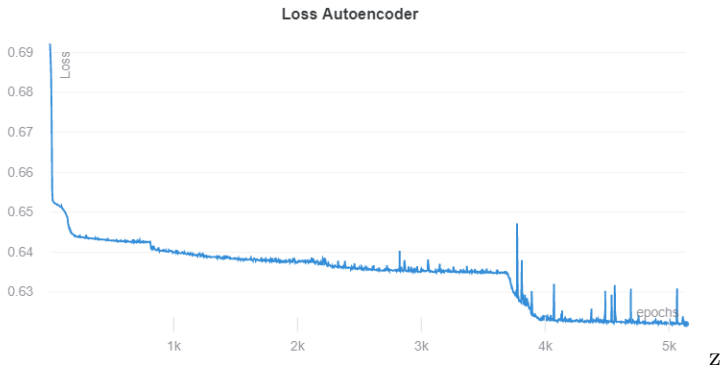


Figure 12: autoencoder training

10 Conclusion

In conclusion different models have been explored and tested. For intra classification the scored achieved is 100% whereas for cross is test 1 equal to 100%, test 2 equal to 50% and test 3 equal to 83%. Unfortunately the score of test 2 for cross did never go beyond 75%. All models are provided in different python files as well as the preprocessing file and the datagenerator file. In the appendix different pictures are present, showing the different random search and the models implemented.

A Appendix

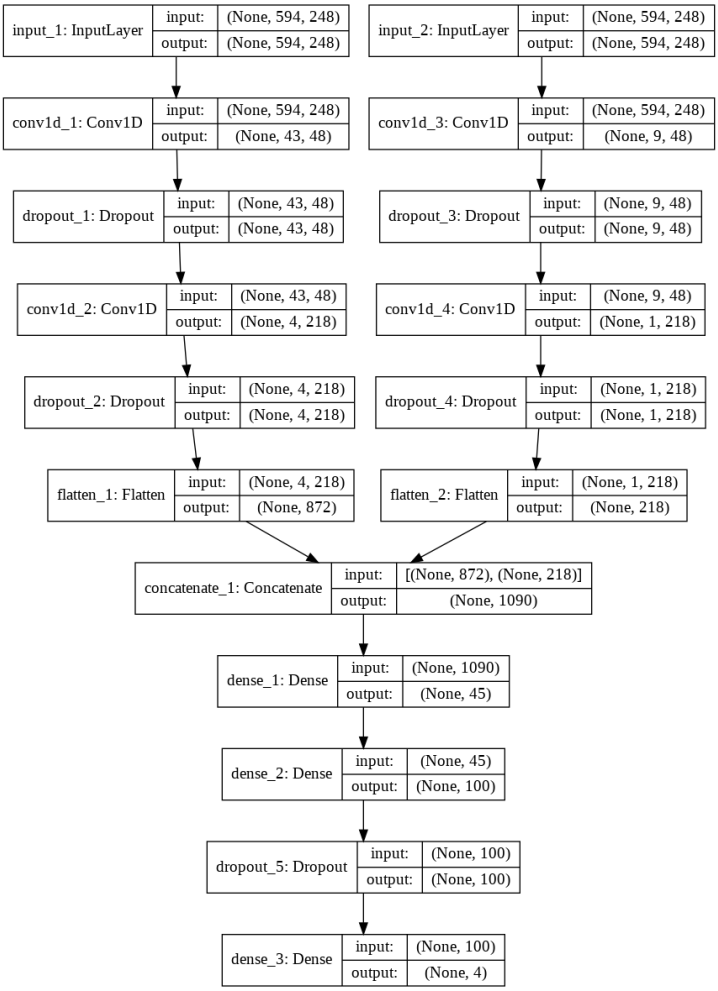


Figure 13: Multi head model

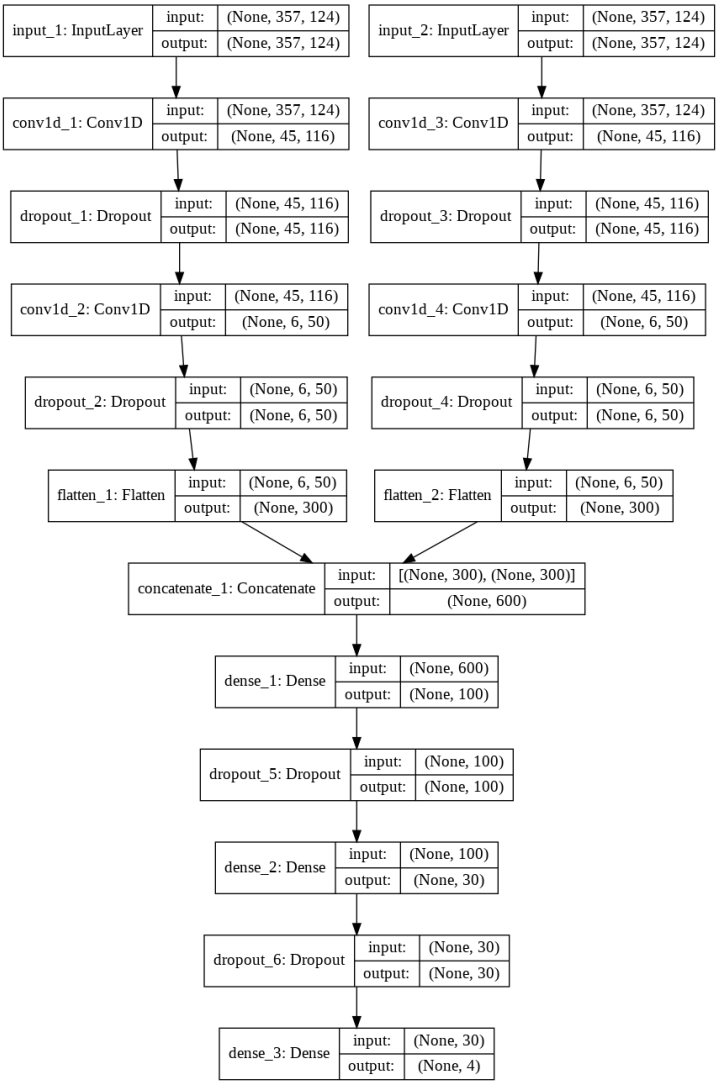


Figure 14: Multi input model

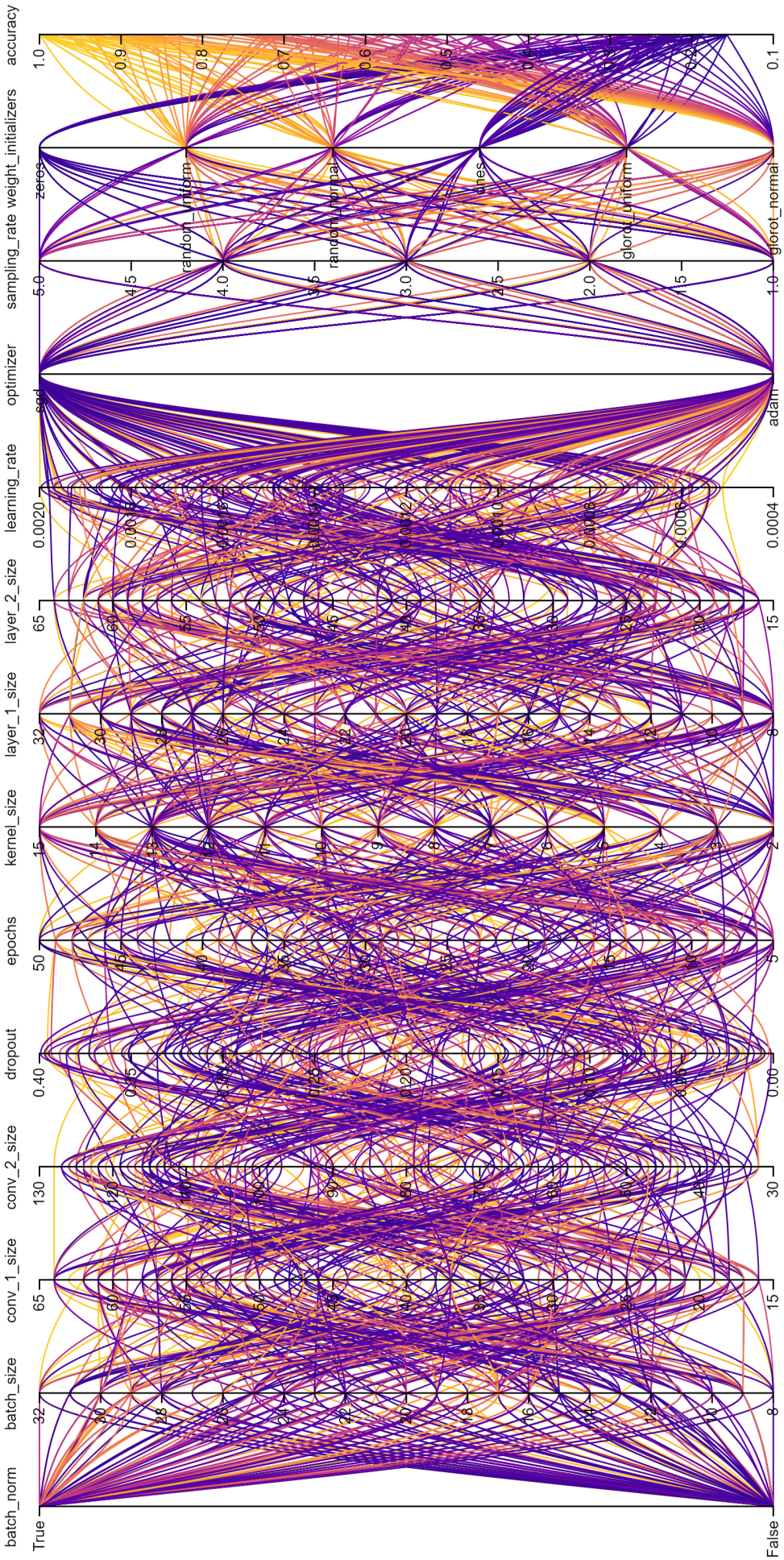


Figure 15: Random search Intra using baseline model

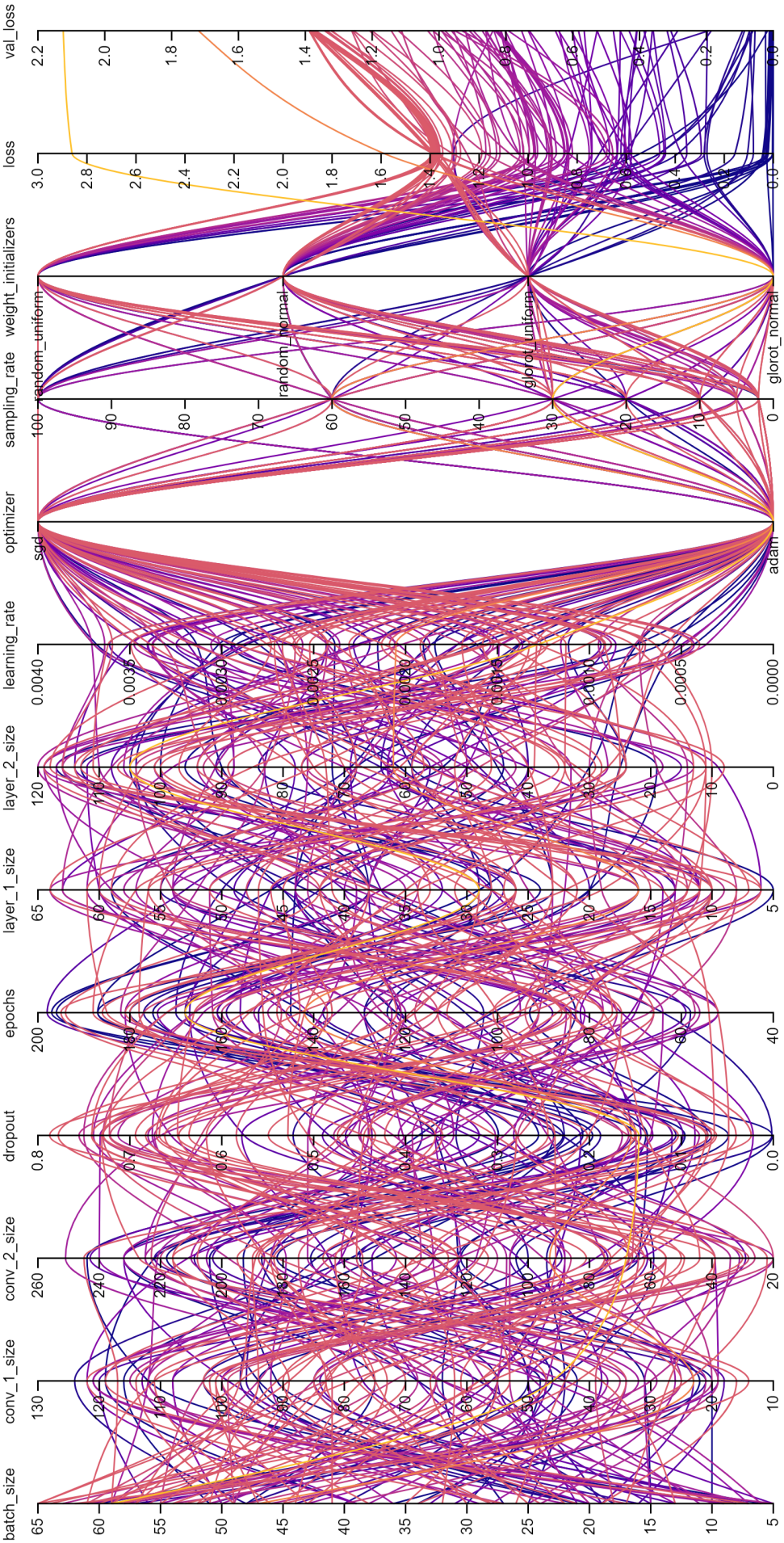


Figure 16: Random search Cross using baseline model

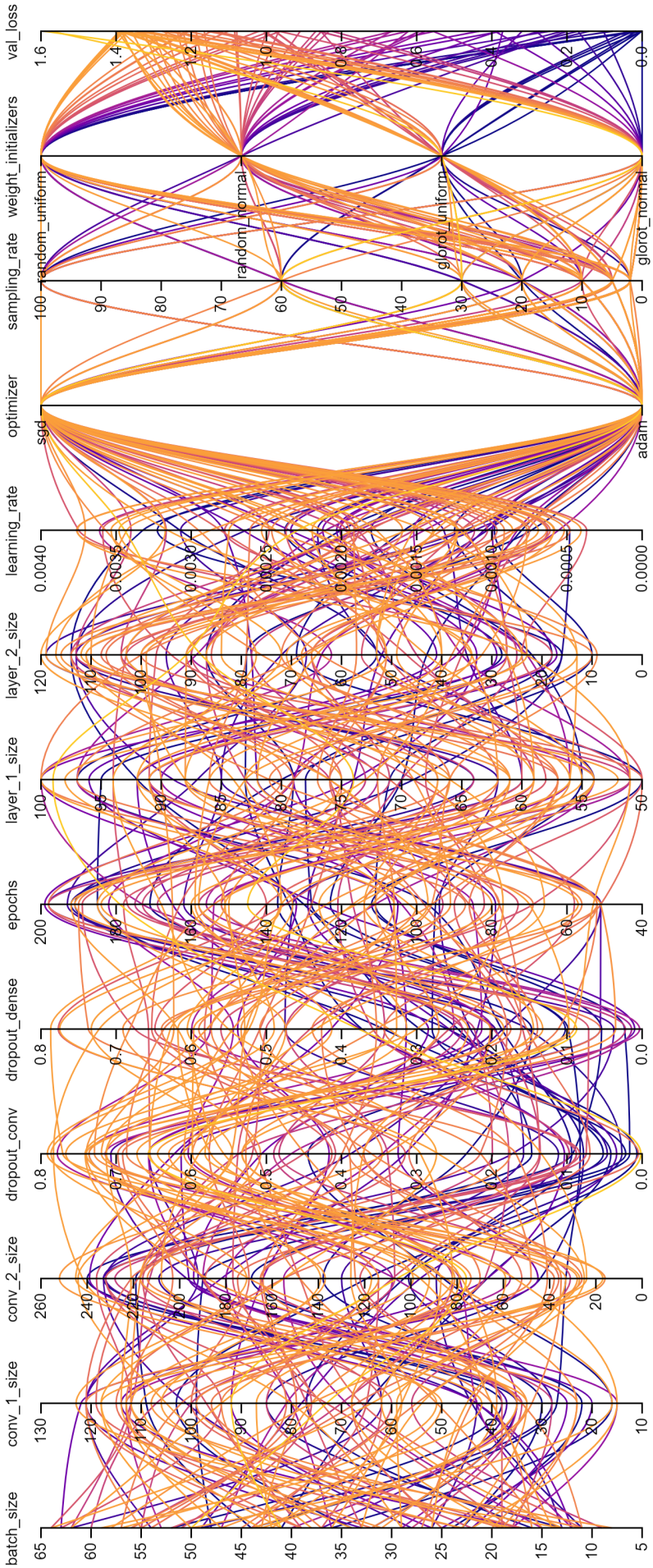


Figure 17: Random search Cross using multi head model