The Algonauts Project 2021

How the Human Brain Makes Sense of a World in Motion [1]

Michal Nahlik (nahlik.michal@seznam.cz)

Code: https://github.com/michal-nahlik/algonauts-vid2fmri-2021

Data preprocessing
Provided videos have length of 3 seconds but with frames per second varying from 15 to 30. To avoid padding and different input lengths all videos were resampled to 5 FPS and the number of frames per second of the original video was used as an input for the model. Adding FPS value as input had positive impact on the model performance while higher FPS or not resampling videos did not. All videos were also resized to 224*224 pixels. To avoid overfitting random number of frames was skipped before re-sampling to make sure different frames are sampled each time and following augmentations were applied: Shift Scale Rotate, Random Perspective change, Coarse Dropout, Random Brightness.

Training
To increase the number of samples for training all possible combinations were generated from the 3 provided repetitions resulting into 7 samples for each video per participant. This greatly improved the ability of the model to generalize and resulted into better score. Output of the model was validated on mean values over the repetitions to follow the method of final evaluation.

First 900 videos from the training dataset were used for training and last 100 for validation. Training was done for 4 epochs with the following setup: Adam optimizer with default parameters for Model A; AdamW with 0.2 weight decay for Model B; PyTorch OneCycleLR scheduler (max LR = 0.001; div_factor=10; final_div_factor=1; pct_start=0.1). Using different optimizers for each model introduced more variety into the ensemble and resulted into small score improvement.

Weighed mean square error was used as loss function:

$$WMSE(y, \hat{y}, W) = \frac{\sum_{i=1}^{N} W_i * (y_i - \hat{y}_i)^2}{N},$$

where y is the target value, ŷ the output of the model and W is the weight of the sample. Weight was calculated for each voxel sample based on the absolute target activity as follows

$$W(y) = \frac{1}{\exp(-|y|)} - 1$$

and clamped to minimum value of 0 and maximum 2

$$f(W) = \begin{cases} 2, W \geq 2 \\ W, 0 < W < 2 \\ 0, W \leq 0 \end{cases}.$$

1 R. M. Cichy, K. Dwivedi, B. Lahner, A. Lascelles, P. Iamshchinina, M. Graumann, A. Andonian, N. A. R. Murty, K. Kay, G. Roig, & A. Oliva. (2021). The Algonauts Project 2021 Challenge: How the Human Brain Makes Sense of a World in Motion.

The target voxel activity has normal distribution so using mean square error loss function without the weights resulted in most activity being predicted around zero. Giving more weight to the samples with higher absolute voxel activity reduced the negative impact of target value imbalance.

Different models were trained for each participant and each track predicting activity of all their voxels simultaneosly.

Models

Each frame of input video was processed by pretrained model from PyTorch Image Models library [2]. The parameters of the pretrained model were frozen so they were not updated during the training. For Model A eca_nfnet_l0 was used and for Model B resnet50 as a backbone model. Features from each layer of the backbone model were pooled, concatenated and used as input for a Linear layer to create final features for each frame (512 for Model A, 1024 for Model B).

Using features from all layers of frozen model resulted in better score than using any single one of them separately, or even unfreezing the parameters and fine tuning the backbone model.

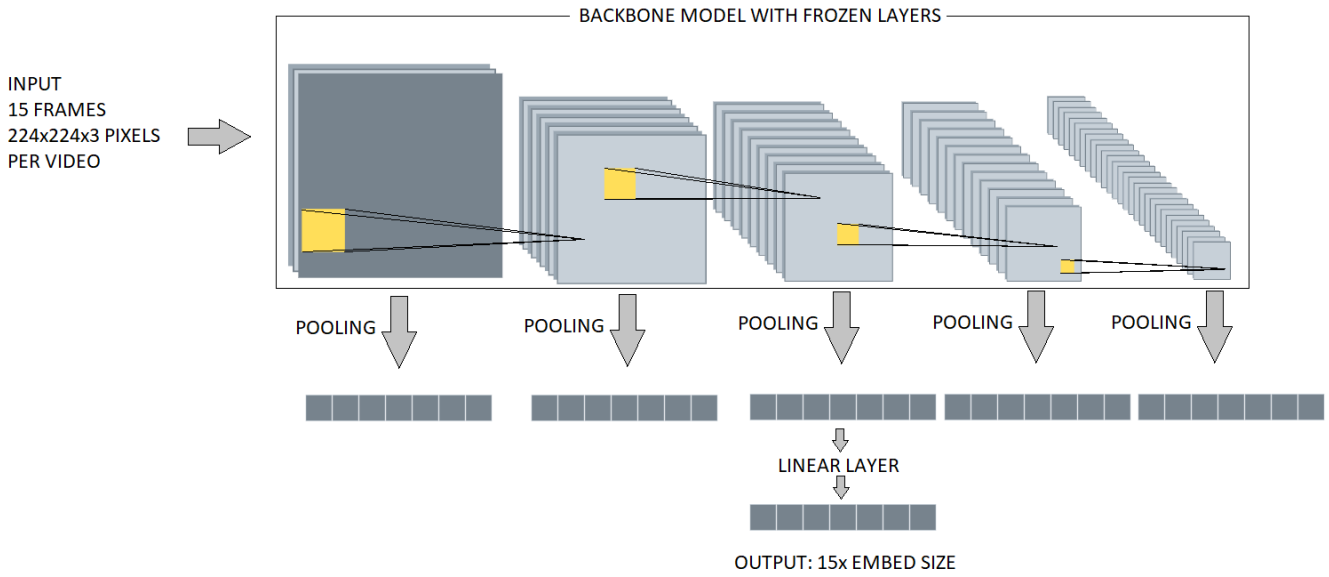

Figure 1: Video encoder. Each frame of video is processed by frozen pretrained model. Features from all layers are pooled, concatenated and used as input to create final features. Output has sequence length (15) with either 512 (Model A) or 1024 (Model B) features.

Model A used maximum and average pooling (1 output value per feature per pooling method). Model B included additional pooling using Linear layers. Output of each backbone feature layer [3] was flattened (resulting in feature space [64, 12544]; [256, 3136]; [512, 784]; [1024, 196]; [2048, 49]) and used as input for 6 Linear layers per each block [4]. In total Model B pooled 8 values (6 using Linear layers, 1from maximum pooling and 1 from average pooling) for each feature (3,904 in total).

2 https://github.com/rwightman/pytorch-image-models
3 total 5, with feature sizes: [64, 112, 112]; [256, 56, 56]; [512, 28, 28]; [1024, 14, 14]; [2048, 7, 7]
4 nn.Linear(12544, 6); nn.Linear(3136, 6); nn.Linear(784, 6); nn.Linear(196, 9); nn.Linear(49, 6)

Minimum, maximum, average and standard deviation was calculated for each feature over the sequence and concatenated with values from last frame and original FPS value of the video. These features (5*512 + 1) were then used to predict voxel response to input video in Model A (Figure 2).

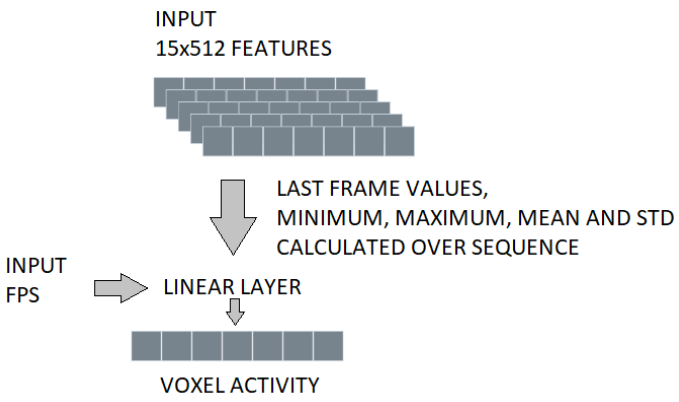

Figure 2: Model A Feature decoder. Features from last frame, values calculated over the sequence (minimum, maximum, mean, std) and original video FPS are used to predict voxel activity.

Model B contains additional block with 3 GRU modules (hidden size 512, layers: 1, 2, 4) that take generated sequence of features as input and add the last output from sequence of each GRU module to features used for voxel activity prediction (Figure 3).

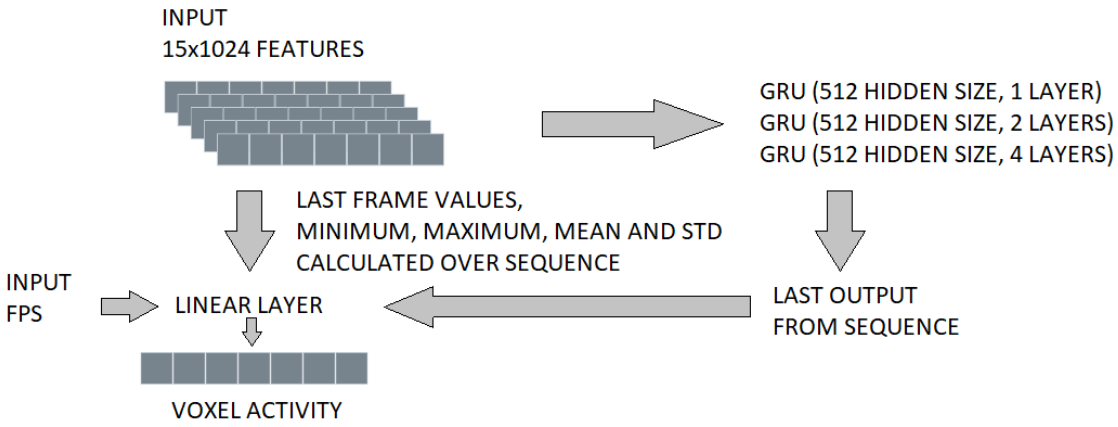

Figure 3: Model B Feature decoder. Features from last frame, values calculated over the sequence (minimum, maximum, mean, std), original video FPS and last output from 3 GRU modules are used to predict voxel activity.

All linear layers in encoder and decoders are using weight normalization. [5] [6]

5 https://pytorch.org/docs/stable/generated/torch.nn.utils.weight_norm.html
6 https://arxiv.org/abs/1602.07868

	Model A	Model B
Backbone (features per layer)	eca_nfnet_l0 (64; 256; 512; 1536; 2304)	Resnet50 (64; 256; 512; 1024; 2048)
Feature pooling	Max (1), Avg (1); (total 9,344 features per frame)	Max (1), Avg (1), Linear (6); (total 31,232 features per frame)
Frame embed size	512	1,024
Sequence features	Last frame, minimum, maximum, mean, standard deviation; (total 2,560 features per video)	Last frame, minimum, maximum, mean, standard deviation, last output of 3 GRU modules (512 hidden size, layers 1, 2, 4); (total 6,656 features per video)
Final features	Sequence features, FPS	Sequence features, FPS
Optimizer	Adam	AdamW (weight decay = 0.2)

Table 1: Model comparison

Prediction, model selection and ensembling

During prediction test data were preprocessed the same way as validation data. Videos were resampled to 5 FPS and resized to 224 * 224 pixels. No additional test time augmentations were used.

For Mini Track models from last training epoch were used to create prediction on test data, while for Full Track the models with best validation score were used.

Final prediction was calculated as average output of Model A and B for each participant and track.

Results

	Score	LOC	FFA	STS	EBA	PPA	V1	V2	V3	V4
Baseline		0.170	0.175	0.090	0.191	0.115	0.200	0.183	0.166	0.139
Model A	**0.253**	0.295	0.269	0.176	0.312	0.208	0.208	0.203	0.202	0.184
Model B	**0.235**	0.258	0.228	0.143	0.280	0.177	0.242	0.236	0.227	0.200
A + B ensemble	**0.266**	0.299	0.267	0.172	0.317	0.206	0.255	0.250	0.243	0.217

Table 2: Mini Track validation results calculated as mean score over 10 participant. For validation last 100 videos from training dataset were used.

	Score
Baseline	**0.069**
Model A	**0.131**
Model B	**0.125**
A+ B ensemble	**0.139**

Table 3: Full Track validation results calculated as mean score over 10 participants. For validation last 100 videos from training dataset were used.

	Score	LOC	FFA	STS	EBA	PPA	V1	V2	V3	V4
Baseline	**0.420**	0.439	0.504	0.332	0.444	0.348	0.444	0.434	0.405	0.428
Model A	**0.522**	0.626	0.698	0.476	0.616	0.553	0.397	0.398	0.436	0.498
A + B ensemble	**0.571**	0.647	0.707	0.484	0.647	0.550	0.499	0.502	0.522	0.580

Table 4: Mini Track test results

	Score
Baseline	**0.206**
A+ B ensemble	**0.312**

Table 5: Full Track test results

Discussion and further improvements

Model A performed better for higher-level regions (LOC, FFA, STS, EBA, PPA), while Model B had better score in early and mid-level visual cortex (V1, V2, V3, and V4). The difference was caused by pooling using Linear layers, simple max and avg pool does not seem to capture features important for ROIs V1 to V4. GRU modules in Model B had minimal impact on score but were kept to introduce more variety into ensemble.

Adding number of frames per second as a feature had positive impact on the prediction accuracy and seems to be important feature.

Both models easily overffited to training data so reducing the number of parameters or using methods like Mean teacher [7] or Stochastic Weight Averaging [8] could be useful for better generalization.

Esembling models led to significant score improvement. Training models in cross validation manner, using test time augmentations or adding models with different backbones and pooling methods could improve score even further.

7 https://arxiv.org/abs/1703.01780
8 https://arxiv.org/abs/1803.05407