BBScore: A Framework for Brain and Behavioral Benchmarking

Evaluating Deep Learning Models Against Biological Neural And Behavioral Data

CS 375: Large-Scale Neural Network Models for Neuroscience

2026.02.17

Atlas Kazemian
Khaled Jedoui Al-Karkari

Stanford Neuro AI Lab
Stanford Neuroscience and Artificial Intelligence Lab

Agenda

1. Motivation: Why benchmark models against the brain?
2. What is BBScore? Pipeline overview
3. Datasets & Benchmarks: Neural and behavioral data
4. Models: 60+ architectures across vision, video, language
5. Metrics: How we measure model-brain alignment
6. Using BBScore: Installation, running, and extending

Motivation

Why benchmark models against the brain?

The Central Question:

How well do deep neural networks learn representations similar to those in biological brains?

Why This Matters

Understanding vision: DNNs represent the best current computational theories of the ventral visual stream.

Model selection: Which architectures best predict neural responses?

Neuroscience insights: What computational principles do brains and DNNs share?

AI improvement: Brain-aligned models often generalize better

The Encoding Model Approach

1. Show same stimuli to model and brain
2. Extract features from model layers
3. Learn a mapping from model features to neural responses
4. Measure prediction accuracy (Pearson r)
5. Compare across models, layers, brain areas

What is BBScore?

A modular framework for benchmarking deep learning models against neural and behavioral datasets.

Data Handling

Standardized loading of diverse neural datasets (fMRI, ephys, behavioral)

Model Integration

60+ pretrained models (CNNs, ViTs, video, language, multimodal)

Feature Extraction

Extract activations from any model layer with aggregation support

Metric Computation

21 metrics for measuring model-brain alignment (ridge, RSA, online, ...)

The BBScore Pipeline

Stimulus Dataset
Images, Videos, or Text

> > >

Model
60+ pretrained architectures

> > >

Feature Extractor
Layer activations + aggregation

> > >

Metric
Ridge, RSA, PLS, Online, ...

> > >

Score
Pearson r, R-squared, ...

Neural Assembly (Brain Data)

fMRI voxels, spiking neurons, or behavioral labels that serve as the prediction target.

```
# Run a benchmark in one command:
python run.py \
    --model resnet50 \
    --layer _orig_mod.resnet.encoder.stages.3 \
    --benchmark NSDV1Shared \
    --metric ridge
```

Datasets & Benchmarks

Neural and behavioral data for evaluating model representations

Available Benchmarks

Benchmark	Type	Species	Modality	Stimuli	Scale
NSD (V1-V4, streams)	Neural	Human	7T fMRI	Natural images	~1,000 images, 4 subjects
TVSD (V1, V4, IT)	Neural	Macaque	Electrophysiology	Object images	22,248 images, 2 monkeys
BMD (22 ROIs)	Neural	Human	3T fMRI	3-sec videos	1,102 videos, 10 subjects
Natural Stories	Neural	Human	3T fMRI	Narrative speech	27 stories, 8 subjects
Physion (Contact)	Behavioral	Human	Behavioral	Physics videos	Binary contact detection
Physion (Placement)	Behavioral	Human	Behavioral	Physics videos	256-class placement
SSV2	Behavioral	Human	Behavioral	Action videos	40 action classes
V1 Sine Gratings	Synthetic	N/A	Synthetic	Sine gratings	Parametric orientations

Natural Scenes Dataset (NSD)

Human fMRI responses to natural images

Species: Human (4 subjects: subj01, subj02, subj05, subj07)

Scanner: 7T fMRI, 1.8 mm isotropic voxels

Stimuli: ~1,000 shared natural images from MS COCO

Brain areas: V1, V2, V3, V4 (dorsal/ventral), Lateral, Ventral, Parietal streams

19 benchmark variants covering individual areas and pathways

Ceiling: Noise-corrected SNR (NCSNR) per voxel, threshold 0.2

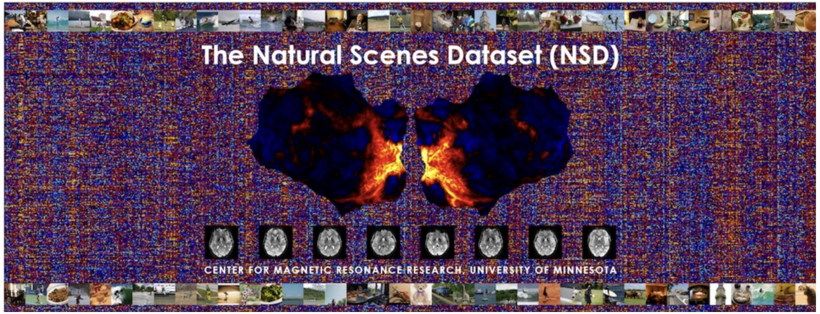
Available NSD Benchmarks

Benchmark	Brain Area
NSDV1Shared	V1 (primary visual)
NSDV2dShared / NSDV2vShared	V2 dorsal / ventral
NSDV3Shared / NSDV4Shared	V3 / V4
NSDVentralShared	Ventral stream
NSDLateralShared	Lateral stream
NSDParietalShared	Parietal stream

Run Commands for Students

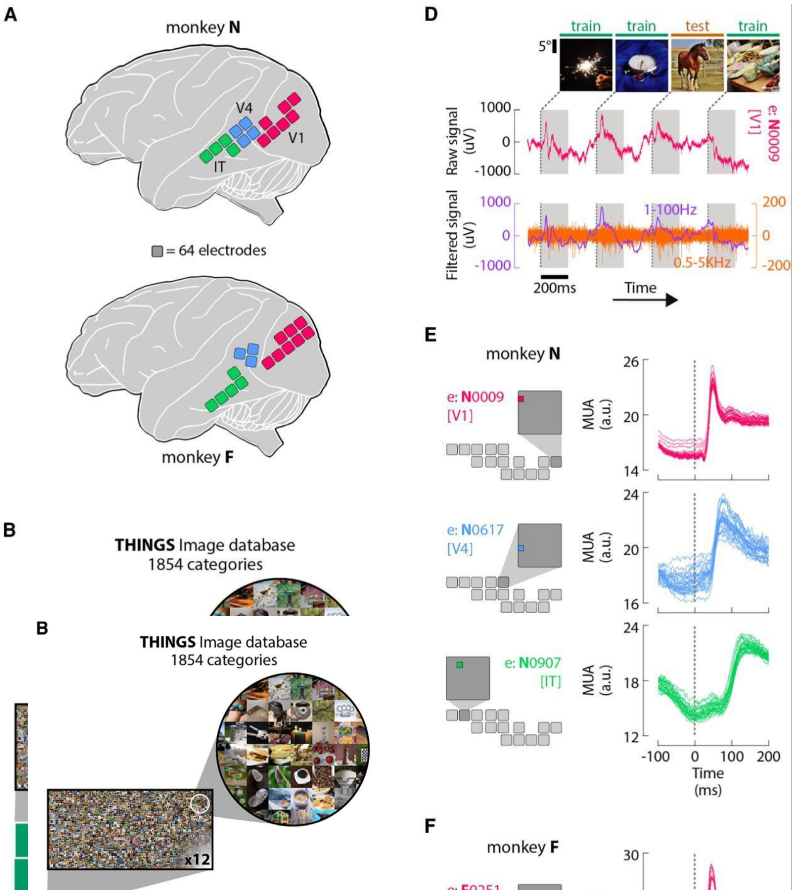
```
# V1 with ResNet-18 (small, CPU-friendly)
python run.py --model resnet18 \
  --layer _orig_mod.resnet.encoder.stages.3 \
  --benchmark NSDV1Shared --metric ridge
```

```
# V4 with DINOv2 (GPU recommended)
python run.py --model dinov2_base \
  --layer blocks.11 \
  --benchmark NSDV4Shared --metric ridge
```



THINGS Ventral Spiking Dataset (TVSD)

Macaque electrophysiology across the ventral stream



Species: 2 macaque monkeys (Monkey F: 1024 units, Monkey N: 1024 units)

Recording: Multi-unit activity (MUA), high-density electrode arrays

Brain areas: V1 (512 ch), V4, IT (inferior temporal) per monkey

Stimuli: 22,248 images from THINGS database (1,854 object categories, 12 imgs each)

Test set: 100 images x 30 repetitions for reliable ceiling estimates

Temporal resolution: Millisecond-level; 10ms, 25ms, 50ms, 100ms bins available

Cross-area mappings: V1-to-V4, V1-to-IT, V4-to-IT (both directions)

Cross-monkey mappings: MonkeyF-to-MonkeyN and reverse for each area

68 benchmark variants (areas, monkeys, time bins, mapping directions)

THINGS Ventral Spiking Dataset (TVSD)

Macaque electrophysiology across the ventral stream

Key Benchmark Names

Benchmark	Type	Description
TVSDV1 / TVSDV4 / TVSDIT	Offline	Single area, ridge
OnlineTVSDV1 / V4 / IT	Online	Streaming, GPU
TVSDV110msBins	Offline	10ms temporal bins
TVSDMonkeyFV1toNV1	Cross-monkey	See next slide

Run Commands for Students

```
# V1 offline (CPU-friendly)
python run.py --model resnet18 \
  --layer _orig_mod.resnet.encoder.stages.0 \
  --benchmark TVSDV1 --metric ridge
```

```
# V4 online (GPU, streaming)
python run.py --model dinov2_base \
  --layer blocks.11 \
  --benchmark OnlineTVSDV4 --metric online_linear_regressor
```

```
# IT with video model (GPU)
python run.py --model videomae_base \
  --layer encoder.layer.11 \
  --benchmark OnlineTVSDIT --metric online_linear_regressor
```

TVSD: Macaque-to-Macaque Cross-Mapping

Can a model's representation map between two different brains?

Instead of model -> brain, these benchmarks test: model features trained on Monkey F -> predict Monkey N (and vice versa). This measures how well a model captures shared neural representations across individuals.

Same Area, Cross-Monkey

Train on Monkey F's V1
Predict Monkey N's V1

Tests: shared representations within the same brain area

Cross-Area, Same Monkey

Train on Monkey F's V1
Predict Monkey F's V4 or IT

Tests: hierarchical transform within one individual

Cross-Area, Cross-Monkey

Train on Monkey F's V1
Predict Monkey N's V4 or IT

Tests: generalization of hierarchical structure

Naming Convention

`TVSDMonkey[F/N][Source]to[F/N][Target]`
`Example: TVSDMonkeyFV1toNV4 = Monkey F V1 -> Monkey N V4`

Run Commands

```
# Same area cross-monkey: F's V1 -> N's V1
python run.py --model None \
  --benchmark TVSDMonkeyFV1toNV1 --metric ridge

# Cross-area cross-monkey: F's V4 -> N's IT
python run.py --model None \
  --benchmark TVSDMonkeyFV4toNIT --metric ridge
```

BOLD Moments Dataset (BMD)

Human fMRI responses to short video clips

Species: 10 human subjects

Scanner: 3T Siemens Trio, 2.5 mm, TR=1.75s

Stimuli: 1,102 three-second video clips from Moments in Time

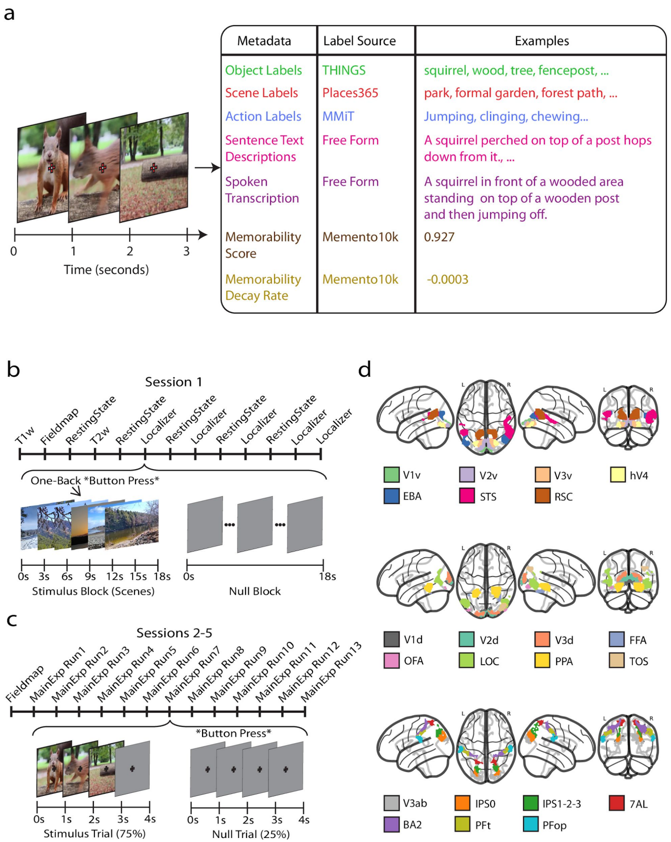
Training: 1,000 videos (3 reps), Test: 102 videos (10 reps)

22 brain ROIs spanning the full visual hierarchy:

Early: V1v/d, V2v/d, V3v/d, V4, V3ab

Category-selective: EBA, FFA, OFA, LOC, PPA, TOS, STS, RSC

Parietal: IPS0, IPS1-2-3, 7AL, BA2, PFt, PFop



Available BMD Benchmarks (22 ROIs)

```
BMD_V1   BMD_V2   BMD_V3   BMD_V4   BMD_V3ab
BMD_FFA   BMD_EBA   BMD_OFA   BMD_LOC   BMD_STS
BMD_PPA   BMD_TOS   BMD_RSC   BMD_MT
BMD_IPS0   BMD_IPS123   BMD_7AL   BMD_BA2   BMD_PFt   BMD_PFop
BMD_V1d   BMD_V1v   BMD_V2d   BMD_V2v   BMD_V3d   BMD_V3v
BMD_BMD (all combined)
```

Lahner et al. (2024) Nature Communications. OpenNeuro ds005165.

BOLD Moments Dataset (BMD)

Human fMRI responses to short video clips

Species: 10 human subjects

Scanner: 3T Siemens Trio, 2.5 mm, TR=1.75s

Stimuli: 1,102 three-second video clips from Moments in Time

Training: 1,000 videos (3 reps), Test: 102 videos (10 reps)

22 brain ROIs spanning the full visual hierarchy:

Early: V1v/d, V2v/d, V3v/d, V4, V3ab

Category-selective: EBA, FFA, OFA, LOC, PPA, TOS, STS, RSC

Parietal: IPS0, IPS1-2-3, 7AL, BA2, PFt, PFop

Available BMD Benchmarks (22 ROIs)

```
BMD_V1   BMD_V2   BMD_V3   BMD_V4   BMD_V3ab
BMD_FFA   BMD_EBA   BMD_OFA   BMD_LOC   BMD_STS
BMD_PPA   BMD_TOS   BMD_RSC   BMD_MT
BMD_IPS0   BMD_IPS123   BMD_7AL   BMD_BA2   BMD_PFt   BMD_PFop
BMD_V1d   BMD_V1v   BMD_V2d   BMD_V2v   BMD_V3d   BMD_V3v
BMD_BMD (all combined)
```

Lahner et al. (2024) Nature Communications. OpenNeuro ds005165.

Run Commands for Students

```
# V1 with video model
python run.py --model videomae_base \
  --layer encoder.layer.11 \
  --benchmark BMD_V1 --metric ridge
```

```
# FFA (face area) with ResNet
python run.py --model resnet50 \
  --layer _orig_mod.resnet.encoder.stages.3 \
  --benchmark BMD_FFA --metric ridge
```

```
# Full BMD (all ROIs) with DINOv2
python run.py --model dinov2_base \
  --layer blocks.11 \
  --benchmark BMD_BMD --metric ridge
```

LeBel et al. 2023: Natural Story Listening Dataset

A natural language fMRI dataset for voxelwise encoding models

Species: 8 human subjects (UTS01-UTS08), ages 21-34

Scanner: 3T Siemens Skyra, 2.6 mm isotropic, TR=2s, 64-ch coil

Stimuli: 27 narrative stories from The Moth Radio Hour podcast

Total: ~370 minutes (6.4h) per subject, 78,893 words

Task: Passive listening to first-person autobiographical stories

Protocol: 5 fMRI sessions, 4-5 training stories + 1 repeated test story each

Brain coverage: Whole cortex; strong responses in temporal, parietal, prefrontal

Use case: Evaluate language models (GPT-2, LLaMA) against brain activity

9 benchmarks in BBScore (default + per-subject: LeBel2023UTS01-UTS08)

```
python run.py \
  --model gpt2_small \
  --layer transformer.h.11 \
  --benchmark LeBel2023 --metric ridge
```

LeBel et al. (2023) Scientific Data. OpenNeuro ds003020.

LeBel et al. 2023: Natural Language fMRI

Voxelwise encoding models for language comprehension

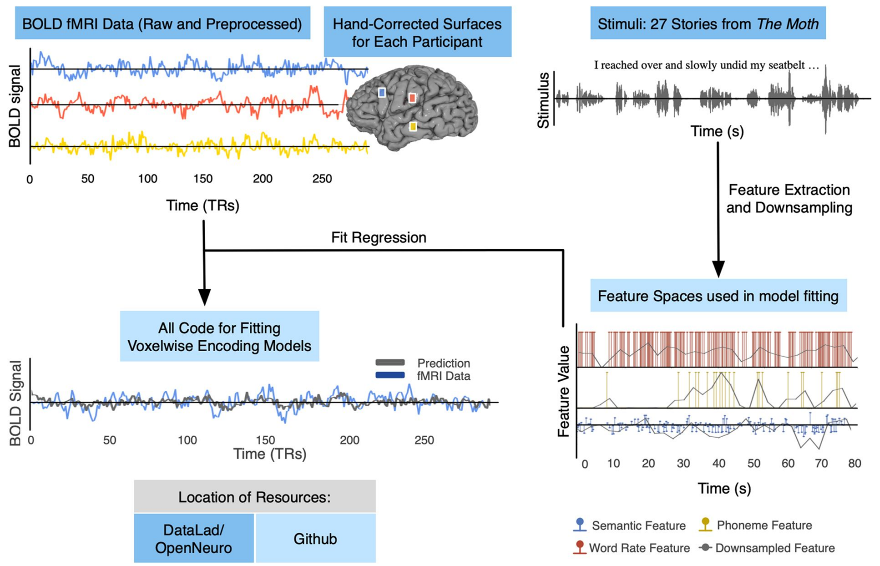
Species: 8 human subjects (UTS01-UTS08), ages 21-34

Scanner: 3T Siemens Skyra, 2.6 mm, TR=2s

Stimuli: 27 narrative stories (~370 min, 78,893 words)

Task: Passive listening to autobiographical stories

Brain coverage: Whole cortex (temporal, parietal, prefrontal)



Story-Averaged Mode

LeBel2023, LeBel2023UTS01...

Averages fMRI across time.
One feature vector per story.
Quick, works with any metric.

TR-Level Mode

LeBel2023TRUTS01...

Preserves temporal structure.
Features per 2s TR window.
HRF delay (4s), GroupKFold CV.
Language mask filtering.

LeBel et al. 2023: Natural Language fMRI

Voxelwise encoding models for language comprehension

Species: 8 human subjects (UTS01-UTS08), ages 21-34

Scanner: 3T Siemens Skyra, 2.6 mm, TR=2s

Stimuli: 27 narrative stories (~370 min, 78,893 words)

Task: Passive listening to autobiographical stories

Brain coverage: Whole cortex (temporal, parietal, prefrontal)

Story-Averaged Mode

LeBel2023, LeBel2023UTS01...

Averages fMRI across time.
One feature vector per story.
Quick, works with any metric.

TR-Level Mode

LeBel2023TRUTS01...

Preserves temporal structure.
Features per 2s TR window.
HRF delay (4s), GroupKFold CV.
Language mask filtering.

LeBel et al. (2023) Scientific Data. OpenNeuro ds003020.

Run Commands for Students

```
# Story-averaged with GPT-2 Small
python run.py --model gpt2_small \
  --layer transformer.h.11 \
  --benchmark LeBel2023 --metric ridge
```

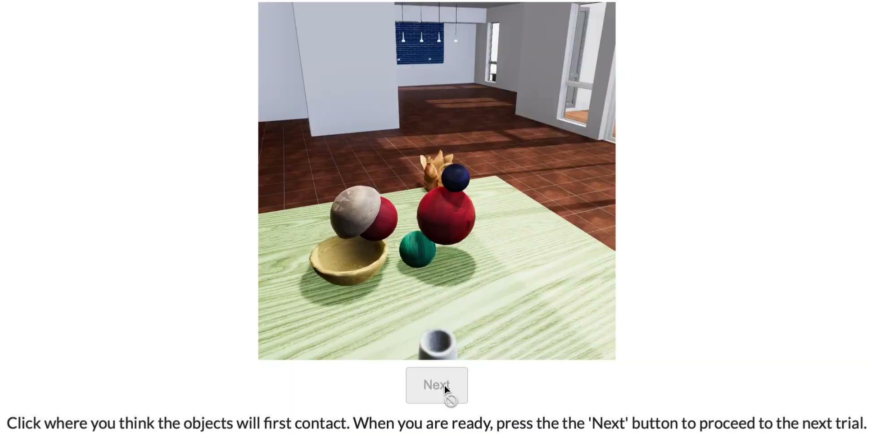
```
# TR-level encoding model (NEW)
python run.py --model gpt2_small \
  --layer transformer.h.11 \
  --benchmark LeBel2023TRUTS01 --metric ridge
```

```
# TR-level temporal RSA
python run.py --model gpt2_small \
  --layer transformer.h.11 \
  --benchmark LeBel2023TRUTS01 --metric temporal_rsa
```

```
# Larger language model
python run.py --model gpt2_large \
  --layer transformer.h.35 \
  --benchmark LeBel2023TRUTS02 --metric ridge
```

Physion: Physical Reasoning

Can models understand intuitive physics?



Contact task: Where will objects first make contact?

Binary classification (contact / no contact)

Detection variant at 10 fps, Prediction variant at 25 fps

Placement task: Where will the object be placed?

256-class classification on a 16x16 grid

Detection and Prediction variants

Human behavioral baselines available on the test set

Intra-distribution variants for within-distribution evaluation

Evaluation metric: Model-human accuracy agreement,

Chebyshev and Euclidean distance on the grid

All online benchmarks using transformer classifiers

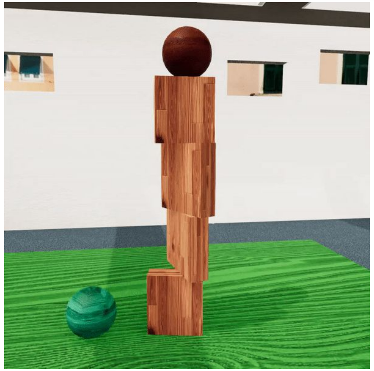
```
OnlinePhysionContactDetection
OnlinePhysionPlacementPrediction
```

Physion: a Diverse Set of Realistic Physical Scenarios

Drop

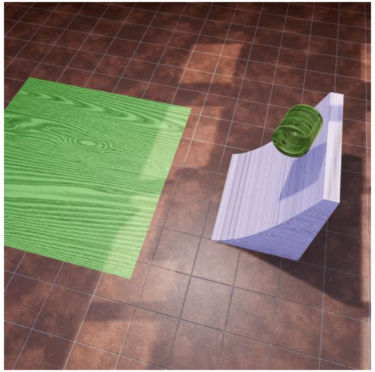
Dominoes

Link



Support

Collide



Roll

Contain

Physion: Physical Reasoning

Can models understand intuitive physics?

Physion Benchmarks

Benchmark	Task	Metric
OnlinePhysionContactDetection	Contact	physion_contact_detection
OnlinePhysionContactPrediction	Contact	physion_contact_prediction
OnlinePhysionPlacementDetection	Placement	physion_placement_detection
OnlinePhysionPlacementPrediction	Placement	physion_placement_prediction

Run Commands for Students

```
# Contact detection with video model
python run.py --model videomae_base \
  --layer encoder.layer.11 \
  --benchmark OnlinePhysionContactDetection \
  --metric physion_contact_detection
```

```
# Placement prediction
python run.py --model videomae_base \
  --layer encoder.layer.11 \
  --benchmark OnlinePhysionPlacementPrediction \
  --metric physion_placement_prediction
```

```
# Contact with DINOv2 (image model on video)
python run.py --model dinov2_base \
  --layer blocks.11 \
  --benchmark OnlinePhysionContactPrediction \
  --metric online_linear_classifier
```

Something-Something V2 (SSV2)

Temporal action recognition from egocentric videos

Task: Classify hand-object interactions from short videos

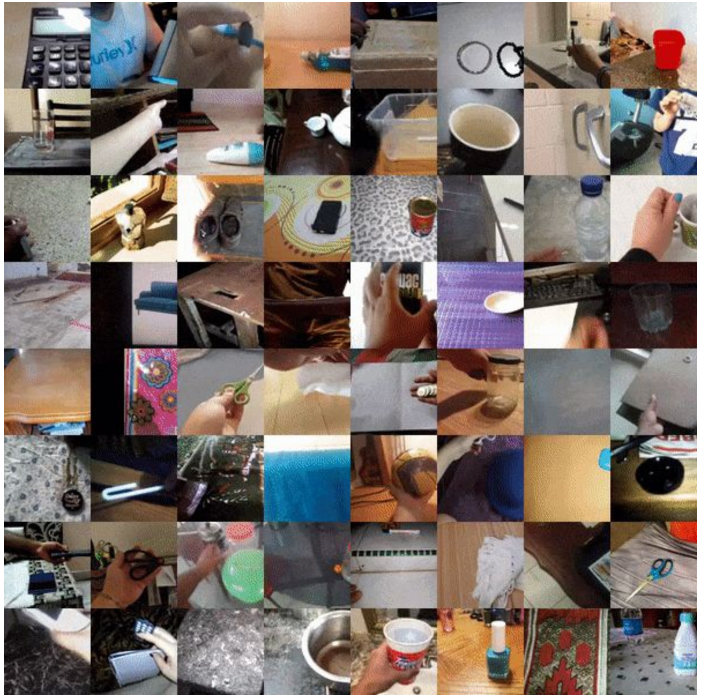
Original dataset: 220K+ videos, 174 action classes

BBScore subset: 40 representative action classes

Target FPS: 12 fps (resampled to consistent frame rate)

Augmented variant: AugmentedSSV2 with RandAugment

Why it matters: Tests temporal dynamics understanding, not just static object appearance



Something-Something V2 (SSV2)

Temporal action recognition from egocentric videos

Run Commands for Students

```
# VideoMAE on SSV2
python run.py --model videomae_base \
  --layer encoder.layer.11 \
  --benchmark SSV2Benchmark \
  --metric online_linear_classifier
```

```
# TimeSformer on SSV2
python run.py --model timesformer_base \
  --layer encoder.layer.11 \
  --benchmark SSV2Benchmark \
  --metric online_linear_classifier
```

```
# Augmented variant with transformer classifier
python run.py --model videomae_base \
  --layer encoder.layer.11 \
  --benchmark AugmentedSSV2Benchmark \
  --metric online_transformer_classifier
```

Models

60+ pretrained architectures for vision, video, language, and more

Models Overview

Category	Count	Examples	Input Type
Classic Vision CNNs	5+	AlexNet, VGG-19, ResNet-18/34/50	Images
Vision Transformers	10+	ViT-B/L/H, Swin-T/S/B/L, ConvNeXt, EfficientNet	Images
Self-Supervised	8+	DINO, DINOv2-B/L/G, MAE-B/L, I-JEPA, MIM	Images
Video Models	15+	VideoMAE, TimeSformer, SlowFast, I3D, X3D, V-JEPA	Video
Language Models	6+	GPT-2 (S/M/L/XL), LLaMA-3-8B, GLM	Text
Multimodal	5+	CLIP (B/32, B/16, L/14), BLIP, S3D-Text	Image+Text
Neuroscience-Inspired	3+	CORnet-Z/RT/S, VOneResNet-50, HMAX	Images
Temporal (LSTM)	5+	ResNet-LSTM, DINOv2-LSTM, MAE-LSTM, R3M-LSTM	Video/Seq
Other Specialized	5+	SAM, R3M, Motion Energy, PixelNeRF, Robust Models	Various

Vision Models: CNNs & Transformers

Convolutional Neural Networks

Model	Params	VRAM	Pretrained On
resnet18	11M	~2 GB	ImageNet
resnet50	26M	~3 GB	ImageNet / DINO
alexnet	61M	~1 GB	ImageNet
vgg19	144M	~4 GB	ImageNet (timm)
convnext_base	89M	~4 GB	ImageNet
efficientnet_b0	5M	~1 GB	ImageNet

Vision Transformers

Model	Params	VRAM	Pretrained On
vit_base	86M	~4 GB	ImageNet
vit_large	304M	~8 GB	ImageNet
swin_base	87M	~4 GB	ImageNet
dinov2_base	86M	~4 GB	Self-supervised
dinov2_large	304M	~8 GB	Self-supervised
clip_vit_b32	151M	~4 GB	Web image-text pairs

Self-Supervised Vision Models

Learning visual representations without labels

DINO / DINOv2	MAE	I-JEPA	V-JEPA
Self-distillation with no labels. Emergent semantic features. Variants: ViT-S/B/L/G. DINOv2: state-of-the-art for many vision tasks.	Masked Autoencoder. Masks 75% of image patches, reconstructs missing pixels. Learns rich visual features from reconstruction.	Image Joint-Embedding Predictive Architecture. Predicts representations of masked regions (not pixels). ViT-H/14 and ViT-g/16.	Video extension of JEPA. Self-supervised video understanding. Predicts masked spatiotemporal tubes in latent space.

Key insight: Self-supervised models often achieve competitive or superior neural alignment compared to supervised models, suggesting the brain may also learn through prediction rather than explicit categorization.

Video Models

Architectures that process temporal sequences

Model	Architecture	Frames	FPS	Pretrained
videomae_base/large	ViT + masked video	16	6.25	Self-supervised
timesformer_base	Divided space-time attention	8	32	Kinetics-400
slowfast_r50	Dual slow/fast pathways	32	12.5	Kinetics-400
i3d_r50	Inflated 3D convolutions	32	12.5	Kinetics-400
x3d_m	Efficient 3D CNN	16	30	Kinetics-400
uniformer_v2_b/l	Conv + Transformer hybrid	8/32	varies	CLIP + K400
vjepa_large/huge	Video JEPA (predictive)	16	10	Self-supervised
cornet_s	Recurrent (neuro-inspired)	var	var	ImageNet
predrnn	ConvLSTM for prediction	var	10	KITTI

LSTM wrappers (resnet_lstm, dinov2_lstm, mae_lstm, r3m_lstm) add temporal recurrence to any static image model via bidirectional LSTM over frame features.

Language, Multimodal & Neuroscience-Inspired Models

Language Models

GPT-2: 4 sizes (124M to 1.5B)

LLaMA-3-8B: Meta's open LLM

Input: tokenized text sequences

Postprocessing: token-to-word alignment

Used with **LeBel2023** benchmark

Multimodal Models

CLIP: Vision + language joint space

ViT-B/32, B/16, L/14, L/14@336

BLIP: Captioning, VQA, matching

Base, Large, VQA, ITM variants

S3D-Text: Video + text model

Vision encoder extracted for benchmarking

Neuroscience-Inspired

CORnet: Recurrent feedback loops

Cortical feedback dynamics

Z (fast), RT, S (standard)

VOneResNet: Gabor front-end

V1-like filters + ResNet-50

HMAX: Hierarchical MAX pooling

Simple/complex cell hierarchy

Metrics

How we measure alignment between model representations and brain data

Metrics Overview

21 metrics across 5 categories

Category	Metrics	GPU?	Description
Regression	ridge, torch_ridge, torch_lasso, torch_elastic, pls	Optional	Linear mapping: model features -> brain responses
Similarity (RSA)	rsa, temporal_rsa, versa	No	Compare representational geometry (RDMs)
Matching	one_to_one, semi_matching, soft_matching, bidirectional	No	Feature-to-feature correspondence
Online Learning	online_linear_regressor, online_linear_classifier, online_transformer_classifier	Yes	Streaming SGD-based training for large data
Specialized	orientation_selectivity, physion_contact_*, physion_placement_*, temporal_ridge, inverse_ridge	Varies	Task-specific: V1 tuning, physics, temporal

Ridge Regression: The Default Metric

Linear mapping from model features to neural responses

How It Works

1. Extract features from a model layer for each stimulus
2. Fit linear regression with L2 regularization:

   brain_response = W * model_features + b

3. Cross-validate over regularization strength (alpha)
4. Evaluate on held-out data using Pearson correlation
5. Ceiling-normalize: divide by noise ceiling (NCSNR)

Ridge Variants in BBScore

Metric Key	Backend	Best For
ridge	sklearn (CPU)	Default, small-medium data
torch_ridge	PyTorch (GPU)	Large data (>2B elements)
torch_lasso	PyTorch (GPU)	Sparse feature selection (L1)
torch_elastic	PyTorch (GPU)	Combined L1+L2
temporal_ridge	Chunked	Temporal brain data (3D)
inverse_ridge	Chunked	Brain-to-model decoding

Tip: For high-dimensional features (>5000), use --subsample-features-for-alpha 2000 to speed up alpha search. Install cuML for ~50x speedup on GPU.

RSA & Matching Metrics

Alternative ways to compare representations

Representational Similarity Analysis (RSA)

Compares **representational geometry**, not raw predictions

1. Build RDM (representational dissimilarity matrix) for model
2. Build RDM for brain data
3. Correlate the two RDMs (Spearman, Kendall, cosine)

temporal_rsa: Static model vs time-resolved brain data

versa (VeRSA): Hybrid -- Ridge prediction + RDM comparison

No GPU required. Good for comparing representational structure.

Matching Metrics

one_to_one: Optimal bipartite matching (Hungarian algorithm)

Best 1:1 correspondence between features

semi_matching: Asymmetric matching

Average of best row-wise and column-wise matches

soft_matching: Optimal transport (Sinkhorn)

Probabilistic soft assignments with regularization

bidirectional: Ridge in both directions

Model->Brain AND Brain->Model mappings

Online Metrics

Memory-efficient streaming metrics for large datasets

When datasets are too large to fit all features in memory, online metrics train incrementally from mini-batches using SGD.

Online Linear Regressor

Linear probe trained via SGD with L2 regularization.

Loss options:
- MSE (default)
- Pearson correlation
- CCC (concordance)
- Combined

Best for: neural prediction from streaming video data.

Online Linear Classifier

Linear classification head trained online.

Supports binary and multiclass tasks.

Best for: behavioral classification (SSV2, Physion).

Online Transformer Classifier

Transformer encoder trained online for sequential inputs.

Configurable: embed_dim, num_heads, num_layers.

Best for: complex temporal pattern recognition.

All online metrics support: grid search (LR, weight decay), WSD/cosine schedulers, mixed precision, early stopping, feature normalization.

Choosing a Metric

Decision guide for your project

What are you trying to measure?

Prediction Accuracy (Pearson r, R-squared)

Use ridge (default)

torch_ridge for GPU
pls for dim. reduction
online_linear_regressor for large video datasets

Representational Geometry (RDM)

Use rsa

temporal_rsa for time-resolved
versa for hybrid approach (Ridge + RDM comparison)

Feature Correspondence

Use one_to_one

semi_matching for asymmetric
soft_matching for probabilistic
bidirectional for mutual mapping

Behavioral Task Performance

Use online_linear_classifier or online_transformer_classifier

physion_* for physics tasks
orientation_selectivity for V1 tuning analysis

Recommendation: Start with ridge for neural benchmarks or online_linear_regressor for online/video benchmarks. Add RSA for representational analysis.

Using BBScore

Installation, running benchmarks, and extending the framework

Installation

System Requirements

Resource	Minimum	Recommended
RAM	8 GB	16+ GB
GPU	None (CPU works)	4+ GB VRAM
Disk	50 GB	100+ GB
Python	3.9+	3.10-3.11

No GPU? No problem!

- Use ridge metric (CPU-only)
- Use smaller models (resnet18)
- Use Online benchmarks (OnlineTVSD*)
- Install CPU-only: ./install.sh --quick --cpu-only

Quick Install (Recommended)

```
# Interactive installer (auto-detects GPU)
chmod +x install.sh
./install.sh

# Or quick install:
./install.sh --quick
```

Manual Install

```
conda create -n bbscore python=3.10 -y
conda activate bbscore
pip install torch torchvision torchaudio
conda install -c conda-forge decord -y
pip install -r requirements.txt

# Set data directory (50GB+ free space)
export SCIKIT_LEARN_DATA="/path/to/data"
```

Running Your First Benchmark

Basic Command Structure

```
python run.py --model <MODEL> --layer <LAYER> --benchmark <BENCHMARK> --metric <METRIC>
```

Example Commands

```
# Start small (CPU-friendly, offline benchmark)
python run.py \
  --model resnet18 \
  --layer _orig_mod.resnet.encoder.stages.3 \
  --benchmark NSDV1Shared \
  --metric ridge
```

```
# Scale up (GPU, online benchmark + online metric)
python run.py \
  --model dinov2_base \
  --layer blocks.11 \
  --benchmark OnlineTVSDV4 \
  --metric online_linear_regressor
```

Common Options & More Examples

Useful Command-Line Options

Option	Default	Description
--batch-size N	4	Adjust based on GPU memory (lower = less memory)
--device cuda:0	auto	Force specific GPU or cpu
--debug	off	Enable detailed logging
--subsample-features-for-alpha N	off	Speed up ridge alpha search for high-dim features
--aggregation-mode	none	Multi-layer: none, concatenate, or stack
--random-projection	off	Dimensionality reduction: dense or sparse

More Examples

```
# Human fMRI (NSD)
python run.py --model resnet50 \
  --layer _orig_mod.resnet.encoder.stages.3 \
  --benchmark NSDV1Shared --metric ridge
```

```
# Multi-layer extraction
python run.py --model dinov2_base \
  --layer blocks.5 blocks.11 \
  --benchmark NSDV4Shared --metric ridge \
  --aggregation-mode concatenate
```

Finding Layer Names

How to discover which layers to extract from

```
from models import MODEL_REGISTRY

# Get the model class
model_info = MODEL_REGISTRY['resnet18']
model_instance = model_info['class']()
model = model_instance.get_model('ResNet18')

# Print all layer names
for name, module in model.named_modules():
    print(name)
```

HuggingFace ResNet Layer Name Mapping

Standard PyTorch	HuggingFace Name
layer1	_orig_mod.resnet.encoder.stages.0
layer2	_orig_mod.resnet.encoder.stages.1
layer3	_orig_mod.resnet.encoder.stages.2
layer4	_orig_mod.resnet.encoder.stages.3

Common Layer Names

Model	Layer Name
ResNet (HF)	_orig_mod.resnet.encoder.stages.0-3
DINOv2	blocks.0 through blocks.11
ViT	encoder.layer.0 through encoder.layer.11
VideoMAE	encoder.layer.0 through encoder.layer.11
TimeSformer	encoder.layer.0 through encoder.layer.11
GPT-2	transformer.h.0 through transformer.h.11
CLIP ViT-B/32	vision_model.encoder.layers.0-11

Extending BBScore

Adding new models, benchmarks, metrics, and datasets

BBScore is designed to be extended. Each component follows the same pattern: create a class, inherit from the base, register it.

New Model

1. Create models/<name>/
2. Implement wrapper class:
   - __init__(): model mappings
   - preprocess_fn(): input prep
   - get_model(): load weights
   - postprocess_fn(): reshape
3. Register in __init__.py:

```
MODEL_REGISTRY["name"] = {
  "class": MyModel,
  "model_id_mapping": "id"
}
```

New Benchmark

1. Create benchmarks/<Name>/
2. Subclass BenchmarkScore or OnlineBenchmarkScore
3. Define in __init__:
   - stimulus_train_class
   - assembly_class
   - train/test kwargs
4. Register in benchmarks/__init__.py BENCHMARK_REGISTRY

New Metric

1. Create metrics/<name>.py
2. Subclass BaseMetric or OnlineMetric
3. Implement:
   - compute_raw(source, target, test_source, test_target)
   - Return dict of scores
4. Register in METRICS dict in metrics/__init__.py

New Dataset

1. Create data/<Name>.py
2. Subclass BaseDataset
3. Implement:
   - __init__(): fetch/extract
   - __len__(): dataset size
   - __getitem__(): load item
4. For neural data:
   - get_assembly(): returns (data, ceiling)
5. Supports S3, GCS, HTTP, Google Drive

Tips & Troubleshooting

Common Issues & Solutions

Out of GPU Memory:

--batch-size 2 or --device cpu --metric ridge

Out of RAM:

Use Online* benchmarks instead of standard ones

Slow Ridge Training:

--subsample-features-for-alpha 2000

Install cuML for ~50x speedup

Dataset Download Issues:

Check SCIKIT_LEARN_DATA is set and has free disk space

Useful Commands

```
# Check your system
python check_system.py --quick

# Detailed check for a specific config
python check_system.py \
  --model resnet50 \
  --benchmark TVSDV1 \
  --metric ridge

# List all available components
python check_system.py --list

# Batch evaluation across configs
python eval.py

# Activate environment
source activate_bbscore.sh
```

Summary

BBScore is a modular framework for comparing deep learning models to biological neural data

Datasets span human fMRI (NSD, BMD, LeBel2023), macaque electrophysiology (TVSD), and behavioral tasks (Physion, SSV2, V1 Gratings)

60+ models covering vision CNNs, transformers, self-supervised, video, language, multimodal, and neuroscience-inspired architectures

21 metrics from simple ridge regression to online transformer classifiers

Easy to use: one command runs the full pipeline

Get started for your final project:

```
./install.sh --quick && source activate_bbscore.sh && python run.py --model resnet18 --layer
_orig_mod.resnet.encoder.stages.3 --benchmark NSDV1Shared --metric ridge
```

Questions?

Good luck with your final projects!

Stanford

Neuro AI Lab