BCILattice

Components Reference Guide

NeuralFlow · MLFlow (ML Suite) · Workflow Designer

v1.0 · BCINexus Platform · 2026-05-20

About This Document

This reference covers every component available in BCILattice's three designer environments: NeuralFlow (paradigm/timeline designer), MLFlow — the ML Suite canvas — and the Workflow Designer. The document is organised in three parts. Each component entry lists its purpose, configurable parameters, and input/output ports.

Contents

Part 1 — NeuralFlow Block Types

NeuralFlow is BCILattice's timeline-based paradigm designer. Blocks are arranged left-to-right on a canvas to define the temporal sequence of a BCI experiment. Each block represents one epoch type, a control marker, or a structural element (Loop Block). Experiments are compiled to a .nflow binary or JSON export. Compiled paradigms drive the Data Manager's epoch labelling during import.

Block Properties

Every block has three editable properties in the Property Inspector:
 • Label — human-readable name (e.g. 'Rest_1'). Displayed on the block and in the timeline summary.
 • Label Code — integer sent to EEG acquisition hardware / LSL stream as the epoch marker. Special blocks have reserved codes (see table below).
 • Duration — epoch length in seconds (range 1–600 s). Not shown for Loop Block (its duration is computed from sub-blocks).

Special Codes (reserved — cannot be changed by user)

Block	Code	Meaning
Start Trial	-1	Start-of-trial marker — signals trial onset to hardware
End Trial	-2	End-of-trial marker — signals trial end to hardware
Marker (LSL/TTL)	-3	Generic LSL or TTL trigger event
Loop Block	-4	Loop container block — duration is auto-computed

State

Block Name	Default Code	Color	Description
Rest	0	#3d5a80	Baseline/rest epoch. Subject is at rest with no task. Used as reference class.
Activity	0	#98c1d9	General activity epoch. Subject performs an unspecified active task.

Motor Imagery

Block Name	Default Code	Color	Description
MI Left Hand	1	#e63946	Motor imagery of left hand movement. Standard 4-class MI paradigm class.
MI Right Hand	2	#457b9d	Motor imagery of right hand movement. Standard 4-class MI paradigm class.
MI Feet	3	#2a9d8f	Motor imagery of feet movement. Standard 4-class MI paradigm class.
MI Tongue	4	#e9c46a	Motor imagery of tongue movement. Standard 4-class MI paradigm class.

Motor Action

Block Name	Default Code	Color	Description
Walk	0	#f4a261	Active walking task. Used in gait/locomotion BCI experiments.
Run	0	#e76f51	Running or fast movement task.
Grip	0	#264653	Hand grip / squeeze action block.
Jump	0	#2a9d8f	Jump action block for lower-limb research.

Cognitive

Block Name	Default Code	Color	Description
Attention	0	#6a0572	Sustained attention task epoch.
Memory	0	#ab83a1	Working memory task epoch (e.g., n-back).
Relaxation	0	#5f7c8a	Relaxation / mindfulness epoch, often used as a baseline alternative to Rest.
Math Task	0	#cb4335	Mental arithmetic task — often used in SSVEP or ERP paradigms.
Visual Search	0	#1a5276	Visual attention / search paradigm epoch.
Mental Rotation	0	#117a65	Mental rotation cognitive task epoch.

Cue / Feedback

Block Name	Default Code	Color	Description
Visual Cue	0	#f39c12	Visual stimulus/cue presented to subject. Triggers epoch onset.
Auditory Cue	0	#8e44ad	Auditory beep or tone presented as a cue.
Tactile Cue	0	#1abc9c	Vibrotactile or other haptic cue block.
Olfactory Cue	0	#e67e22	Olfactory (smell) stimulus — rare BCI modality.
Visual Feedback	0	#2ecc71	Closed-loop visual feedback delivered to subject after classification.
Auditory Feedback	0	#3498db	Auditory feedback tone played after classifier output.
Haptic Feedback	0	#e91e63	Haptic/vibrotactile feedback after classification output.

Control

Block Name	Default Code	Color	Description
Start Trial	-1 (reserved)	#ff5252	Special marker block. Emits code -1. Signals start of a trial to acquisition hardware.
End Trial	-2 (reserved)	#ff5252	Special marker block. Emits code -2. Signals end of a trial.
Marker (LSL/TTL)	-3 (reserved)	#ffab40	Sends an LSL event marker or TTL trigger. Code -3. Used for hardware synchronisation.

Utility

Block Name	Default Code	Color	Description
Fixed Time	0	#607d8b	Fixed-duration blank period. No stimulus. Duration configurable. Useful for ITI (inter-trial interval).
Loop Block	-4 (reserved)	#9c27b0	Special container block. Holds sub-blocks with individual repeat counts. Code -4. Duration computed from sub-block sum × repeat.

Loop Block — Detail

The Loop Block is a special container that repeats a set of sub-blocks a configurable number of times. In the Property Inspector, set 'Total Repeat' (how many times the entire loop runs) then add sub-block rows — each row selects an existing block on the canvas and a per-sub-block repeat count. The Loop Block's total duration is computed as sum(sub_block_duration × sub_block_repeat) × loop_total_repeat. Loop Blocks cannot be nested inside other Loop Blocks.

Inspector Fields

Field	Type	Description
Label	str	Display name for the loop container block
Label Code	int	Marker code emitted at loop start — default -4 (reserved)
Total Repeat	int (1–100)	How many times the entire loop repeats
Sub-Block (each row)	dropdown	Select an existing block on the canvas
Sub-Block Repeat	int (1–1000)	Per-sub-block repeat count within one loop iteration

Export Formats

Format	Extension	Notes
NeuralFlow Binary	.nflow	Compact binary format. Fastest load; recommended for local BCILattice sessions.
JSON	.json	Human-readable key-value export. Useful for inspection and version control.
CSV	.csv	Flat table export of the block sequence with labels, codes, and durations.

Part 2 — MLFlow (ML Suite) Components

The ML Suite canvas (MLFlow Designer) is a node-graph pipeline builder. Drag blocks from the left panel onto the canvas, connect output ports to input ports, then Compile and Train. BCILattice includes the following block categories, all supporting both MLFlow and Workflow.

Pipeline Blocks (Built-in / Custom)

These blocks are built into BCILattice.

Input Block

Defines an external input port for the pipeline. When a compiled pipeline is called, data is passed in through this port.

Parameter	Type	Default	Description
port_name	str	In_1	Name of the port, must be unique within the pipeline

Inputs: None | Outputs: Data (Any)

Output Block

Defines an external output port. Data flowing into this block is returned when the pipeline finishes executing.

Parameter	Type	Default	Description
port_name	str	Out_1	Name of the output port

Inputs: Data (Any) | Outputs: None

Loop Block

Repeats a subgraph a fixed number of times. The sub-graph runs for each iteration; loop output is passed forward after all iterations complete.

Parameter	Type	Default	Description
iterations	int	1	Total number of loop iterations
start_node	str		Name of the first block inside the loop
end_node	str		Name of the last block inside the loop
mode	select	Epoch	Sequential — feed all data at once; Epoch — iterate one epoch at a time

Inputs: Loop Input (Any) | Outputs: Loop Output (Any)

Custom Code

Execute arbitrary Python code as a pipeline stage. Write a function named 'custom_process(data)' that returns the transformed data.

Parameter	Type	Default	Description

code	code editor	def custom_process(data):\n return data	Python source code executed at runtime

Inputs: Data (Any) | Outputs: Result (Any)

Analysis (Custom Blocks)

These three blocks live in the Analysis category and are injected by BCILattice, not part of the main block catalog.

Channel Selection

Runs a statistical channel selection algorithm on the input EEG/fNIRS/EMG data and returns a filtered dataset containing only the most discriminative channels. Uses one of several ranking methods (t-Value, ANOVA, etc.) and a percentile threshold.

Parameter	Type	Default	Description
method_name	select	t-Value Method	Channel ranking algorithm
rest_label	int	0	Integer label used for the rest/baseline class
percentile	int	5	Bottom-N percentile of channels to remove (5 = remove worst 5%)
ignore_classes	str		Comma-separated class labels to exclude from scoring

Inputs: Data (Any), Labels (Any) | Outputs: Selected Data (Any), Selected Channels (Any), Scores (Any)

Custom Labels

Attach or override class labels for a dataset. Labels can be loaded from a CSV/TXT file or entered manually as a comma-separated string.

Parameter	Type	Default	Description
label_file	str		Path to a file containing one label per line
manual_labels	str		Comma-separated label list, e.g. '0,1,0,1,2'

Inputs: Data (Any) | Outputs: Labels (Any)

Signal Projection

Converts 1-D time-series data into 2-D image representations suitable for convolutional neural networks. Three methods are available (via the pyts library).

Parameter	Type	Default	Description
method_name	select	Gramian Angular Field (GAF)	Projection method: GAF, Markov Transition Field, or Recurrence Plot
image_size	int	64	Output image size in pixels (square)

Inputs: Data (Any) | Outputs: Projected Images (Any), Shape (Any)

BCI EEG Models

Custom EEG classification architectures built with PyTorch and designed for BCI paradigms (motor imagery, P300, SSVEP). All models accept raw or bandpass-filtered EEG epochs and output a predicted class.

Model	Key Parameters	Description
ATCNet	n_channels=22, n_classes=4, eegnet_F1=16, eegnet_D=2, tcn_depth=2	Attention Temporal Convolutional Network. Combines EEGNet, TCN, and attention for MI.
DeepConvNet	n_channels=64, n_classes=4, n_filters_start=25, dropout=0.5	Deep ConvNet for raw EEG. 4-stage temporal/spatial convolution blocks.
EEGConformer	n_channels=22, n_classes=4, n_filters=40, depth=6, attn_heads=8	Transformer-based EEG model combining CNN feature extraction with self-attention.
EEGNet	n_channels=64, n_classes=4, F1=8, D=2, F2=16	Compact depthwise separable CNN, the standard BCI baseline. Low parameter count.
ShallowConvNet	n_channels=64, n_classes=4, n_filters=40, filter_length=25, pool_length=75	Shallow CNN baseline from Schirrmeister et al. 2017. Strong for oscillatory BCI.

Sequence Models

LSTM, GRU, BiLSTM, Mamba, and TCN architectures for sequential EEG/fNIRS classification.

Block Name	Notes
BiLSTMClassifier	Bidirectional LSTM. Captures forward and backward temporal dependencies.
GRUClassifier	Gated Recurrent Unit — lighter than LSTM, similar accuracy.
LSTMClassifier	Standard LSTM for temporal EEG sequences.
MambaSSM	State Space Model (Mamba). Linear-time sequence modelling — new SOTA option.
TCNClassifier	Temporal Convolutional Network with dilated causal convolutions.

Transformer Models

Transformer-based EEG encoders for large-scale or multi-session BCI applications.

Block Name	Notes
BrainBERT	BERT architecture pre-trained on large EEG corpora.
EEGTransformer	Multi-head self-attention encoder for EEG classification.
PatchTST	Patch-based Transformer for time-series / EEG. Segments signal into patches.
ViTEEG	Vision Transformer applied to 2-D EEG spectrograms.

Edge Models

Lightweight / quantized models optimised for deployment on Raspberry Pi, NVIDIA Jetson, and other edge hardware. Exported as ONNX or TFLite.

Block Name	Notes
MobileEEGNet	MobileNet-style depthwise-separable EEGNet variant — < 50 K params.
QuantizedEEGNet	INT8-quantized EEGNet for MCU / Jetson deployment.
TinyEEGNet	Ultra-compact EEGNet for extreme edge targets.

GPT Models

Large language / generative pre-trained model backbones adapted for EEG classification and BCI.

Block Name	Notes
BioGPTBCI	BioGPT backbone fine-tuned for BCI token classification.
GPT2BCI	GPT-2 (gpt2 / gpt2-medium / gpt2-large) adapted for EEG.
LaBraMBCI	LaBraM large-scale BCI foundation model.
LargeGPT	Large GPT variant for research experiments.
MiniGPT	Compact GPT for low-resource deployment.
TinyGPT	Smallest GPT variant — suited for edge inference.

Feature Selection

Statistical and model-based methods to select the most informative features before classification.

Block	Type
GenericUnivariateSelect	Function
RFE	Class
RFECV	Class
SelectFdr	Function
SelectFpr	Function
SelectFromModel	Class
SelectFwe	Function
SelectKBest	Function
SelectPercentile	Function
SelectorMixin	Class
SequentialFeatureSelector	Class
VarianceThreshold	Class
chi2	Function
f_classif	Function
f_oneway	Function
f_regression	Function
mutual_info_classif	Function
mutual_info_regression	Function
r_regression	Function

Cross Validation

Splitters, search strategies (GridSearchCV, RandomizedSearchCV), and threshold classifiers.

Block	Type
BaseCrossValidator	Class
BaseShuffleSplit	Class
FixedThresholdClassifier	Class
GridSearchCV	Function
GroupKFold	Class
GroupShuffleSplit	Class
KFold	Function
LearningCurveDisplay	Class
LeaveOneGroupOut	Class
LeaveOneOut	Class
LeavePGroupsOut	Class
LeavePOut	Class
ParameterGrid	Function
ParameterSampler	Function
PredefinedSplit	Class
RandomizedSearchCV	Function
RepeatedKFold	Function
RepeatedStratifiedKFold	Class
ShuffleSplit	Function
StratifiedGroupKFold	Function
StratifiedKFold	Class
StratifiedShuffleSplit	Class
TimeSeriesSplit	Class
TunedThresholdClassifierCV	Class
ValidationCurveDisplay	Class
check_cv	Function
cross_val_predict	Function
cross_val_score	Function
cross_validate	Function
permutation_test_score	Function
train_test_split	Function
validation_curve	Function

Dimensionality Reduction

PCA, ICA, NMF, LDA, and other matrix factorisation methods for reducing feature dimensionality.

Block	Type
AlignedUMAP	Class
ClassicalMDS	Class
DictionaryLearning	Class
FactorAnalysis	Class
FastICA	Class
IncrementalPCA	Class
Isomap	Class
KernelPCA	Class
LatentDirichletAllocation	Class

LocallyLinearEmbedding	Class
MDS	Class
MiniBatchDictionaryLearning	Class
MiniBatchNMF	Class
MiniBatchSparsePCA	Function
NMF	Class
PCA	Class
ParametricUMAP	Function
SparseCoder	Class
SparsePCA	Function
SpectralEmbedding	Class
TSNE	Class
TruncatedSVD	Class
UMAP	Class
dict_learning	Function
dict_learning_online	Function
fastica	Function
locally_linear_embedding	Function
non_negative_factorization	Function
randomized_svd	Function
smacof	Function
sparse_encode	Function
spectral_embedding	Function
trustworthiness	Function

ML Linear

Linear classifiers and regressors: Logistic Regression, SVM, Ridge, Lasso, ElasticNet, etc.

Block	Type
ARDRegression	Class
BayesianRidge	Class
ElasticNet	Class
ElasticNetCV	Class
GammaRegressor	Function
HuberRegressor	Class
Lars	Class
LarsCV	Class
Lasso	Function
LassoCV	Class
LassoLars	Function
LassoLarsCV	Function
LassoLarsIC	Class
LinearRegression	Class
LogisticRegression	Class
LogisticRegressionCV	Class
MultiTaskElasticNet	Class
MultiTaskElasticNetCV	Class
MultiTaskLasso	Function
MultiTaskLassoCV	Class
OrthogonalMatchingPursuit	Class
OrthogonalMatchingPursuitCV	Class
PassiveAggressiveClassifier	Class
PassiveAggressiveRegressor	Class
Perceptron	Function

PoissonRegressor	Function
QuantileRegressor	Class
RANSACRegressor	Class
Ridge	Class
RidgeCV	Class
RidgeClassifier	Class
RidgeClassifierCV	Class
SGDClassifier	Class
SGDOneClassSVM	Class
SGDRegressor	Function
TheilSenRegressor	Class
TweedieRegressor	Function
enet_path	Function
lars_path	Function
lars_path_gram	Function
lasso_path	Function
orthogonal_mp	Function
orthogonal_mp_gram	Function
ridge_regression	Function

ML Ensemble

Ensemble methods: Random Forest, Extra Trees, Gradient Boosting, XGBoost-like wrappers, AdaBoost, Bagging.

Block	Type
AdaBoostClassifier	Class
AdaBoostRegressor	Class
BaggingClassifier	Class
BaggingRegressor	Class
BaseDecisionTree	Class
BaseEnsemble	Function
Booster	Class
DMatrix	Class
DataIter	Class
DecisionTreeClassifier	Class
DecisionTreeRegressor	Class
ExtMemQuantileDMatrix	Function
ExtraTreeClassifier	Function
ExtraTreeRegressor	Function
ExtraTreesClassifier	Function
ExtraTreesRegressor	Function
GradientBoostingClassifier	Class
GradientBoostingRegressor	Class
HistGradientBoostingClassifier	Class
HistGradientBoostingRegressor	Class
IsolationForest	Class
QuantileDMatrix	Function
RabitTracker	Class
RandomForestClassifier	Function
RandomForestRegressor	Function
RandomTreesEmbedding	Class
StackingClassifier	Class
StackingRegressor	Class

VotingClassifier	Class
VotingRegressor	Class
XGBClassifier	Class
XGBModel	Class
XGBRFClassifier	Class
XGBRFRegressor	Class
XGBRanker	Class
XGBRegressor	Function
build_info	Function
config_context	Function
cv	Function
export_graphviz	Function
export_text	Function
get_config	Function
plot_importance	Function
plot_tree	Function
plot_tree	Function
set_config	Function
to_graphviz	Function
train	Function

ML Neighbors

k-Nearest Neighbours classifiers, regressors, and distance structures (KDTree, BallTree).

Block	Type
BallTree	Function
KDTree	Function
KNeighborsClassifier	Class
KNeighborsRegressor	Class
KNeighborsTransformer	Class
KernelDensity	Class
LinearSVC	Class
LinearSVR	Class
LocalOutlierFactor	Class
NearestCentroid	Class
NearestNeighbors	Class
NeighborhoodComponentsAnalysis	Class
NuSVC	Function
NuSVR	Function
OneClassSVM	Class
RadiusNeighborsClassifier	Class
RadiusNeighborsRegressor	Class
RadiusNeighborsTransformer	Class
SVC	Function
SVR	Function
kneighbors_graph	Function
l1_min_c	Function
radius_neighbors_graph	Function
sort_graph_by_row_values	Function

ML Clustering

Unsupervised clustering: K-Means, DBSCAN, Agglomerative, Birch, Mean-Shift, and more.

Block	Type
AffinityPropagation	Class
AgglomerativeClustering	Class
BayesianGaussianMixture	Function
Birch	Class
BisectingKMeans	Class
DBSCAN	Class
FeatureAgglomeration	Class
GaussianMixture	Class
HDBSCAN	Class
KMeans	Class
MeanShift	Class
MiniBatchKMeans	Class
OPTICS	Class
SpectralBiclustering	Function
SpectralClustering	Class
SpectralCoclustering	Function
affinity_propagation	Function
cluster_optics_dbscan	Function
cluster_optics_xi	Function
compute_optics_graph	Function
dbscan	Function
estimate_bandwidth	Function
get_bin_seeds	Function
k_means	Function
kmeans_plusplus	Function
linkage_tree	Function
mean_shift	Function
spectral_clustering	Function
ward_tree	Function

Preprocessing

Scalers (StandardScaler, MinMaxScaler, RobustScaler), encoders, binarisers, and imputers.

Block	Type
Binarizer	Class
FunctionTransformer	Class
KBinsDiscretizer	Class
KNNImputer	Class
KernelCenterer	Class
LabelBinarizer	Class
LabelEncoder	Class
MaxAbsScaler	Class
MinMaxScaler	Class
MissingIndicator	Class
MultiLabelBinarizer	Class
Normalizer	Class
OneHotEncoder	Class
OrdinalEncoder	Class
PolynomialFeatures	Class
PowerTransformer	Class

QuantileTransformer	Class
RobustScaler	Class
SimpleImputer	Class
SplineTransformer	Class
StandardScaler	Class
TargetEncoder	Class
add_dummy_feature	Function
binarize	Function
label_binarize	Function
maxabs_scale	Function
minmax_scale	Function
normalize	Function
power_transform	Function
quantile_transform	Function
robust_scale	Function
scale	Function

Evaluation

Classification, regression, clustering, and ranking metrics plus display utilities.

Block	Primary Use
ConfusionMatrixDisplay	Evaluation metric
DetCurveDisplay	Evaluation metric
DistanceMetric	Evaluation metric
PrecisionRecallDisplay	Evaluation metric
PredictionErrorDisplay	Evaluation metric
RocCurveDisplay	Evaluation metric
accuracy_score	Overall classification accuracy
adjusted_mutual_info_score	Evaluation metric
adjusted_rand_score	Evaluation metric
auc	Evaluation metric
average_precision_score	Evaluation metric
balanced_accuracy_score	Accuracy adjusted for class imbalance
brier_score_loss	Evaluation metric
calinski_harabasz_score	Evaluation metric
check_scoring	Evaluation metric
class_likelihood_ratios	Evaluation metric
classification_report	Full precision/recall/F1 per class
cohen_kappa_score	Inter-rater reliability / agreement
completeness_score	Evaluation metric
confusion_matrix	Per-class confusion table
confusion_matrix_at_thresholds	Evaluation metric
consensus_score	Evaluation metric
coverage_error	Evaluation metric
d2_absolute_error_score	Evaluation metric
d2_brier_score	Evaluation metric
d2_log_loss_score	Evaluation metric
d2_pinball_score	Evaluation metric
d2_tweedie_score	Evaluation metric
davies_bouldin_score	Evaluation metric
dcg_score	Evaluation metric
det_curve	Evaluation metric
euclidean_distances	Evaluation metric
explained_variance_score	Evaluation metric

f1_score	Harmonic mean of precision and recall
fbeta_score	Evaluation metric
fowlkes_mallows_score	Evaluation metric
get_scorer	Evaluation metric
get_scorer_names	Evaluation metric
hamming_loss	Evaluation metric
hinge_loss	Evaluation metric
homogeneity_completeness_v_measure	Evaluation metric
homogeneity_score	Evaluation metric
jaccard_score	Evaluation metric
label_ranking_average_precision_score	Evaluation metric
label_ranking_loss	Evaluation metric
log_loss	Evaluation metric
make_scorer	Evaluation metric
matthews_corrcoef	MCC — best single metric for imbalanced BCI
max_error	Evaluation metric
mean_absolute_error	Evaluation metric
mean_absolute_percentage_error	Evaluation metric
mean_gamma_deviance	Evaluation metric
mean_pinball_loss	Evaluation metric
mean_poisson_deviance	Evaluation metric
mean_squared_error	Regression MSE
mean_squared_log_error	Evaluation metric
mean_tweedie_deviance	Evaluation metric
median_absolute_error	Evaluation metric
multilabel_confusion_matrix	Evaluation metric
mutual_info_score	Evaluation metric
nan_euclidean_distances	Evaluation metric
ndcg_score	Evaluation metric
normalized_mutual_info_score	Evaluation metric
pair_confusion_matrix	Evaluation metric
pairwise_distances	Evaluation metric
pairwise_distances_argmin	Evaluation metric
pairwise_distances_argmin_min	Evaluation metric
pairwise_distances_chunked	Evaluation metric
pairwise_kernels	Evaluation metric
precision_recall_curve	Precision vs recall at varying thresholds
precision_recall_fscore_support	Evaluation metric
precision_score	Evaluation metric
r2_score	Coefficient of determination
rand_score	Evaluation metric
recall_score	Evaluation metric
roc_auc_score	Area under the ROC curve
roc_curve	TPR vs FPR at varying thresholds
root_mean_squared_error	Evaluation metric
root_mean_squared_log_error	Evaluation metric
silhouette_samples	Evaluation metric
silhouette_score	Clustering quality metric
top_k_accuracy_score	Evaluation metric
v_measure_score	Evaluation metric
zero_one_loss	Evaluation metric

Model Interpretation (SHAP)

Explainability toolkit. SHAP (SHapley Additive exPlanations) computes feature importance for any ML model. BCILattice ships 63 SHAP blocks covering all explainer types and visualisation utilities.

Block	Category
ActionOptimizer	SHAP explainability
AdditiveExplainer	SHAP explainability
AlphaSelection	SHAP explainability
ClassBalance	SHAP explainability
ClassPredictionError	SHAP explainability
ClassificationReport	SHAP explainability
ClassificationScoreVisualizer	SHAP explainability
CoalitionExplainer	SHAP explainability
Cohorts	SHAP explainability
ConfusionMatrix	SHAP explainability
CooksDistance	SHAP explainability
DeepExplainer	SHAP explainability
ExactExplainer	SHAP explainability
Explainer	SHAP explainability
Explanation	SHAP explainability
GPUTreeExplainer	SHAP explainability
GradientExplainer	SHAP explainability
KernelExplainer	SHAP explainability
LinearExplainer	SHAP explainability
ManualAlphaSelection	SHAP explainability
PRCurve	SHAP explainability
PartitionExplainer	SHAP explainability
PermutationExplainer	SHAP explainability
PrecisionRecallCurve	SHAP explainability
PredictionError	SHAP explainability
ROCAUC	SHAP explainability
RegressionScoreVisualizer	SHAP explainability
ResidualsPlot	SHAP explainability
SamplingExplainer	SHAP explainability
ScoreVisualizer	SHAP explainability

Note: Showing 30 of 63 SHAP blocks. All are accessible in the ML Suite panel.

Signal Processing — MNE

273 MNE-Python classes and functions for EEG/MEG/fNIRS signal processing. Key block groups:

- Epochs & Events: BaseEpochs, Epochs, EvokedArray, Evoked, Events, ...
- Raw IO: RawArray, RawEDF, RawBDF, RawFIF, RawBrainVision, ...
- Preprocessing: ICA, Covariance, Annotations, ...
- Source Analysis: SourceEstimate, MixedSourceEstimate, VectorSourceEstimate, ...
- Connectivity: AcqParserFIF, ...
- Spatial Filters: Covariance, ...

Note: Full list of 273 MNE blocks available in the ML Suite left panel under 'Signal Processing (MNE)'.

Signal Processing — General

228 blocks from pywt (PyWavelets), scipy, and librosa for wavelets, spectral analysis, and audio feature extraction. Covers: ContinuousWavelet, DiscreteWavelet, dwt, wavedec, pywt tree nodes, scipy spectrogram, stft, hilbert transform, and more.

Data Manipulation

173 NumPy utility functions for array reshaping, concatenation, linear algebra, and mathematical operations. Examples: asarray, concatenate, reshape, transpose, dot, linalg.svd, fft.fft, and statistical functions (mean, std, percentile).

Deep Learning Layers

167 PyTorch layer classes for building custom neural network architectures. Includes: Conv1d, Conv2d, LSTM, GRU, TransformerEncoder, MultiheadAttention, BatchNorm1d, Dropout, Linear, ReLU, Sigmoid, and all pooling, activation, and normalisation layers.

Optimization (PyTorch Optimizers)

16 optimizers for training PyTorch models.

Optimizer
ASGD
Adadelta
Adafactor
Adagrad
Adam
AdamW
Adamax
LBFGS
Muon
NAdam
Optimizer
RAdam
RMSprop
Rprop
SGD
SparseAdam

NLP (NLTK)

46 Natural Language Processing blocks for text preprocessing. Primarily useful in BCI experiments that involve speech or language-related paradigms (e.g., imagined speech BCI). Covers stemmers, tokenisers, taggers, and parsers.

Deep Learning Models (3376 blocks)

The full PyTorch model zoo: AlexNet, ResNet (18/34/50/101/152), VGG, DenseNet, EfficientNet (B0–B7), MobileNet, ViT, ConvNeXt, Swin Transformer, and all weight variants (_Weights enums). Typically used in the Signal Projection pipeline (after converting EEG to 2-D images via the Signal Projection block).

> *Note: Full list available in the ML Suite panel. 3376 blocks across all variants.*

Computer Vision (371 blocks)

Module: cv2 (OpenCV) — Image processing utilities useful when EEG/fNIRS data has been projected to 2-D images. Covers feature detectors (AKAZE, ORB, SIFT), filters, morphological operations, and histogram analysis.

> *Note: Full list available in the ML Suite panel. 371 blocks.*

Math Operations (315 blocks)

Linear algebra routines: matrix decompositions (LU, QR, SVD, Cholesky), eigenvalue solvers, matrix functions, and bandwidth utilities. Useful as intermediate computation nodes in custom pipelines.

> *Note: Full list available in the ML Suite panel. 315 blocks.*

Part 3 — Workflow Designer Components

The Workflow Designer is a higher-level execution graph builder. Unlike MLFlow (which runs a single training pipeline per session), the Workflow Designer orchestrates multi-step processes: file selection → preprocessing → channel selection → pipeline execution → result output, all in a single runnable graph. All MLFlow blocks are available in Workflow plus three additional workflow-only blocks.

Shared with MLFlow: All 24 CSV categories (BCI EEG Models, Feature Selection, Cross Validation, ML Linear, ML Ensemble, Signal Processing, Evaluation, etc.) plus Pipeline Blocks (Input, Output, Loop, Custom Code) and Analysis (Channel Selection, Custom Labels, Signal Projection). See Part 2 for details.

Workflow-Only Custom Blocks

These three blocks are exclusive to the Workflow Designer canvas.

Select Files

Data source selector block. Opens a session file browser and lets the user pick which BCILattice session files or subjects to feed into the workflow. Outputs a list of file paths that downstream blocks consume.

Parameter	Type	Default	Description
modality	select	fNIRS	Signal modality: EEG, fNIRS, or EMG
data_source	select	Clean Data	Which version of the data to load: Clean Data, Channel Selected Data, Features, Dim Reduced Data
session	select	All Sessions	Filter by a specific session name or use All Sessions
selected_files	str		Manually entered file path list (comma-separated)
selected_subjects	str		Comma-separated subject IDs to include

Inputs: None | Outputs: Files (Any)

Graph Endpoint

Exposes the workflow execution as an HTTP API endpoint. When triggered, it runs the connected pipeline graph and returns the result via the configured route. Useful for integrating BCILattice workflows into web applications or external services.

Parameter	Type	Default	Description
endpoint_name	str	workflow_graph	Unique name for this endpoint
route	str	/api/workflow/graph/run	Route path
http_method	select	POST	HTTP method: POST or GET
output_path	str		Optional file path to write response

Inputs: Graph (Any), Files (Any) | Outputs: Response (Any)

Graph Output

Saves a workflow result graph (e.g., accuracy charts, feature importance plots) to disk in a chosen image format.

Parameter	Type	Default	Description
graph_name	str	Workflow Graph	Title label for the saved graph
save_path	str		Output file path (directory or full path)
image_format	select	png	File format: png, jpg, or pdf

Inputs: Graph Data (Any) | Outputs: Graph File (Any)

Typical Workflow Graph Pattern

A complete BCI workflow graph follows this pattern:

- Step 1: Select Files — Choose sessions, modality, and data source
- Step 2: Channel Selection — Rank and remove low-discriminability channels
- Step 3: Signal Processing / Preprocessing — Apply scalers, imputers, or feature extraction
- Step 4: ML Pipeline (via MLFlow block) — Run a compiled MLFlow pipeline against each file
- Step 5: Evaluation — Compute accuracy, confusion matrix, F1 scores
- Step 6: Graph Output — Save result plots to disk
- Step 7: Graph Endpoint (optional) — Expose results via REST API

Component Summary

Designer	Total Blocks	Categories	Notes
NeuralFlow	28	6	Paradigm/timeline blocks only — no ML
MLFlow	5,409+	24 + 2 custom	Built-in block catalog + Pipeline Blocks + Analysis
Workflow	5,412+	24 + 2 custom + 3 WF-only	All MLFlow blocks + Select Files, Graph Endpoint, Graph Output