BNN layers

BayesForge exposes its Bayesian Neural Network building blocks through `m.bnn`. The regression and classification tutorials — BNN for Regression, BNN for classification and BNN for multiclass classification — are built entirely from a single stable building block, `m.bnn.layer_linear`.

The remaining layers documented on this page are **experimental**. They are not used in any of the BNN tutorials yet, their APIs may change, and each prints a development warning at call time. They are provided for advanced users experimenting with structured-covariance and attention architectures.

The stable building block: `layer_linear`

`m.bnn.layer_linear` is the documented layer used across the BNN tutorials. It performs a linear transform `X @ W` (optionally `+ bias`) followed by an optional activation.

```
# W is a prior over the weight array; its shape sets the layer's in/out dimensions
W = m.dist.normal(0, 1, shape=(n_in, n_hidden))
h = m.bnn.layer_linear(X, W, activation='tanh')
```

- `X`: layer input.
- `dist`: prior array for the weights (shape defines input/output dimensions).
- `activation`: one of the names returned by `m.bnn.available_activations()` (e.g. `'relu'`, `'tanh'`, `'sigmoid'`, `'softmax'`), or `None` for a purely linear layer.
- `bias`: an optional bias array, or `False`.

Everything below is **experimental** and is *not* part of these tutorials.

Experimental structured-covariance layers

These layers return a structured, symmetric positive-definite `(block_size, block_size)` covariance block. They are intended for models that learn a structured covariance (e.g. via `m.bnn.cov`) rather than for standard feed-forward prediction.

Diagonal

Independent variables — only per-variable variances are learned.

```
C = m.bnn.layer_diagonal(block_size=8)      # (8, 8) diagonal SPD block
```

Compound symmetry

All variables share a common variance and a common correlation `rho`: `C = sigma^2 * ((1 - rho) I + rho 11^T)`.

```
C = m.bnn.layer_compound_symmetry(block_size=8)
```

Toeplitz

A decaying-correlation structure: correlation falls off with the lag between variables (`sigma * alpha^{|i-j|}`), plus a learned diagonal.

```
C = m.bnn.layer_toeplitz(block_size=8)
```

Common arguments:

- `block_size`: dimension of the square block.
- `sample`: draw the layer's latent parameters directly (`True`) instead of registering them as model sites — used when generating data outside a fit.
- `name`: suffix appended to each internal parameter name (keeps sites unique when a layer is used more than once).
- `seed`: optional PRNG seed for direct `sample=True` draws.

Experimental attention layer: `layer_attention`

`m.bnn.layer_attention` builds a learnable coupling block through a scaled dot-product attention mechanism over learnable query / key-value embeddings. It returns a `(b_q, b_kv)` matrix.

```
block = m.bnn.layer_attention(b_kv=8, b_q=8, d_model=32)
```

- `b_q`, `b_kv`: query- and key/value-block sizes (the output shape is `(b_q, b_kv)`).
- `d_model`: embedding / projection dimension.
- `sample`, `name`, `seed`: as above.

Experimental covariance network: `cov`

`m.bnn.cov` is a small two-layer BNN used to estimate per-unit offsets for a covariance structure. It maps a one-hot encoding of `N` units through a hidden `tanh` layer to two offsets per unit, added to the supplied `a` and `b`, and registers the result as the deterministic site `'rf'`.

```
rf = m.bnn.cov(hidden_dim=16, N=N_units, a=a, b=b)   # deterministic 'rf', shape (2, N)
```

- `hidden_dim`: number of hidden units.
- `N`: number of units (rows of the one-hot input).
- `a`, `b`: base values the network learns offsets around.
- `sample`: draw the weights directly instead of registering them as sites.