Appendix A EvoVis Dashboard

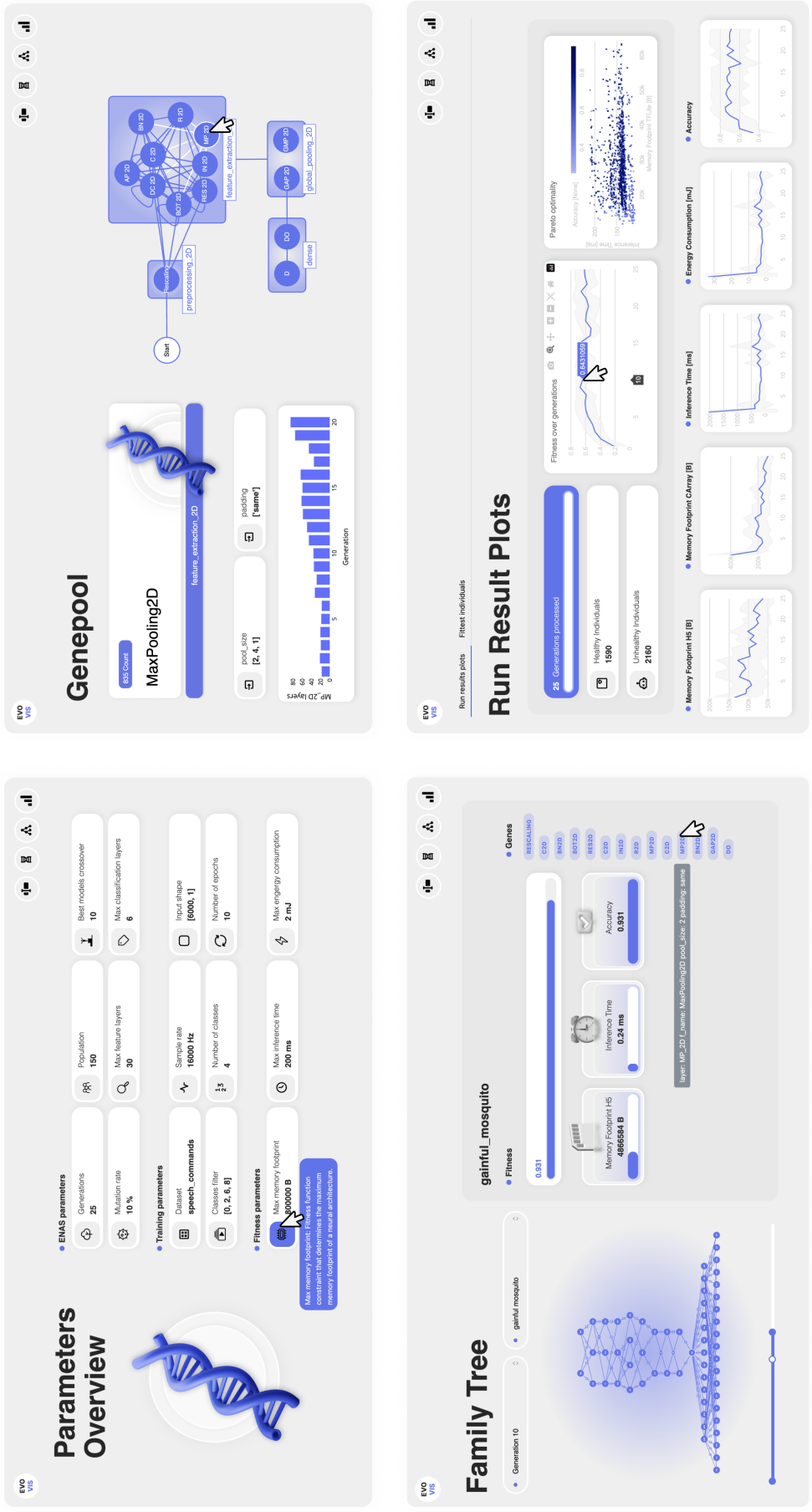

Fig. A1 EvoVis dashboard overview of the different pages showing hyperparameters, search space, family tree and NAS results.

Appendix B NAS Hyperparameters

Listing 1 Example (Speech Commands dataset) for the hyperparameters used in *EdgeVolution*. This file must be defined for each dataset in order to be able to perform the optimization. In the future, it is conceivable that this will be abstracted to a graphical user interface.

```yaml
### General parameters:
min_free_space_gpu:  # Minimum free space in GPU in bytes to start a new training (as we train in parallel)
  value: 6_000_000
limit_tensor_arena_size: # Limit the tensor arena size in bytes (if None, it will use the maximum available tensor arena size)
  value: null

### Dataset/DNN training hyper-parameters:
results_path:
  value: "Results/"
dataset_name:
  value: "speech_commands"
sample_rate:
  value: 6000
input_shape:
  value: [6000, 1]
num_classes:
  value: 12
top_activation:
  value: "softmax"
num_epochs:
  value: 3
batch_size:
  value: 128
optimizer:
  value: "adam"
loss:
  value: "categorical_crossentropy"
metrics:
  value: ["accuracy"]

### EdgeVolution hyper-parameters:
num_generations:
  value: 30
population_size_decay:
  value: [[1, 5], [2, 250], [10, 100]] # [(from_generation, population_size), (from_generation, population_size), ...]
num_best_models_crossover_decay:
  value: [[1, 100], [2, 50], [10, 20]] # [(from_generation, population_size), (from_generation, population_size), ...]
mutation_rate_decay:
  value: [[1, 30], [2, 25], [5, 20], [10, 15]] # [(from_generation, population_size), (from_generation, population_size), ...]
max_num_feature_layers:
  value: 8
max_num_classification_layers:
  value: 4

### Fitness
min_rom_usage:
  value: 150_000
min_energy_information:
  value: 0.5
acc_weight:
  value: 0.7
rom_usage_weight:
  value: 0.1
energy_information_weight:
  value: 0.2
```

Appendix C Search space

Listing 1 Search space (gene pool) configuration for neural architecture search. Values are following either the scheme [”option1”, ”option2”, ”option3”] or [start, stop, step].

```yaml
gene_pool:
  preprocessing_2D:
    - layer: STFT_2D
      f_name: STFT
      n_fft: [64, 512, 16]
      hop_length: [128, 396, 16]
      input_data_format: ['channels_last']
      output_data_format: ['channels_last']

    - layer: MAG_2D
      f_name: Magnitude()

    - layer: FB_2D
      f_name: get_filterbank_layer
      type: ['mel']
      n_mels: [32, 128, 8]
      mel_f_min: [0, 0, 1]
      mel_f_max: [3000, 3000, 1]
      output_data_format: ['channels_last']

    - layer: MAG2DEC_2D
      f_name: MagnitudeToDecibel()

  feature_extraction_2D:
    - layer: C_2D_BLOCK
      f_name: get_conv2d_block
      filters: [4, 64, 1]
      kernel_height: [1, 12, 1]
      kernel_width: [1, 12, 1]
      strides: [1, 2, 1]
      padding: ['same']
      norm_layer: ['None', 'BatchNormalization']
      activation: ['None', 'relu', 'sigmoid', 'tanh', 'leaky_relu']

    - layer: DC_2D_BLOCK
      f_name: get_depthwise_conv2d_block
      kernel_height: [1, 12, 1]
      kernel_width: [1, 12, 1]
      strides: [1, 2, 1]
      padding: ['same']
      norm_layer: ['None', 'BatchNormalization']
      activation: ['None', 'relu', 'sigmoid', 'tanh', 'leaky_relu']

    - layer: C_2D
      f_name: Conv2D
      filters: [4, 48, 1]
      kernel_size: [1, 5, 1]
      strides: [1, 2, 1]
      padding: ['same']
      activation: ['relu']

    - layer: DC_2D
      f_name: DepthwiseConv2D
      kernel_size: [1, 5, 1]
      strides: [1, 2, 1]
      padding: ['same']
      activation: ['relu']

    - layer: MP_2D
      f_name: MaxPooling2D
      pool_size: [2, 4, 1]
      padding: ['same']

    - layer: AP_2D
      f_name: AveragePooling2D
      pool_size: [2, 4, 1]
      padding: ['same']

    - layer: BN_2D
      f_name: BatchNormalization()

  global_pooling_2D:
    - layer: GAP_2D
      f_name: GlobalAveragePooling2D()

    - layer: GMP_2D
      f_name: GlobalMaxPooling2D()

  dense:
    - layer: D
      f_name: Dense
      units: [8, 64, 8]
      activation: ['relu']
```

Listing 2 Rule set that defines how each block of the defined search space is connected with each other.

```yaml
rule_set:
  ################################################
  # Sorted by groups. Each group is a list of layers, where 'rule' is a list of layers that can
  ↪  be used after the current layer.
  ################################################

  Start:
    rule: ["STFT_2D"]

  ################################################
  # 2D layers
  ################################################

  STFT_2D:
    rule: ["MAG_2D"]

  MAG_2D:
    rule: ["C_2D_BLOCK", "DC_2D_BLOCK", "FB_2D", "MAG2DEC_2D"]

  FB_2D:
    rule: ["C_2D_BLOCK", "DC_2D_BLOCK", "MAG2DEC_2D"]

  MAG2DEC_2D:
    rule: ["C_2D_BLOCK", "DC_2D_BLOCK"]

  C_2D_BLOCK:
    rule: ["AP_2D", "MP_2D", "C_2D_BLOCK", "DC_2D_BLOCK"]

  DC_2D_BLOCK:
    rule: ["AP_2D", "MP_2D", "C_2D_BLOCK", "DC_2D_BLOCK"]

  MP_2D:
    rule: ["C_2D_BLOCK", "DC_2D_BLOCK"]

  AP_2D:
    rule: ["C_2D_BLOCK", "DC_2D_BLOCK"]

  GAP_2D:
    rule: ["D"]

  GMP_2D:
    rule: ["D"]

  ################################################
  # Dense layers
  ################################################

  D:
    rule: ["D"]

################################################
# Rule set group just for Dashboard visualization
################################################
rule_set_group:
  - group: "feature_extraction_2D"
    rule: ["global_pooling_2D"]
```

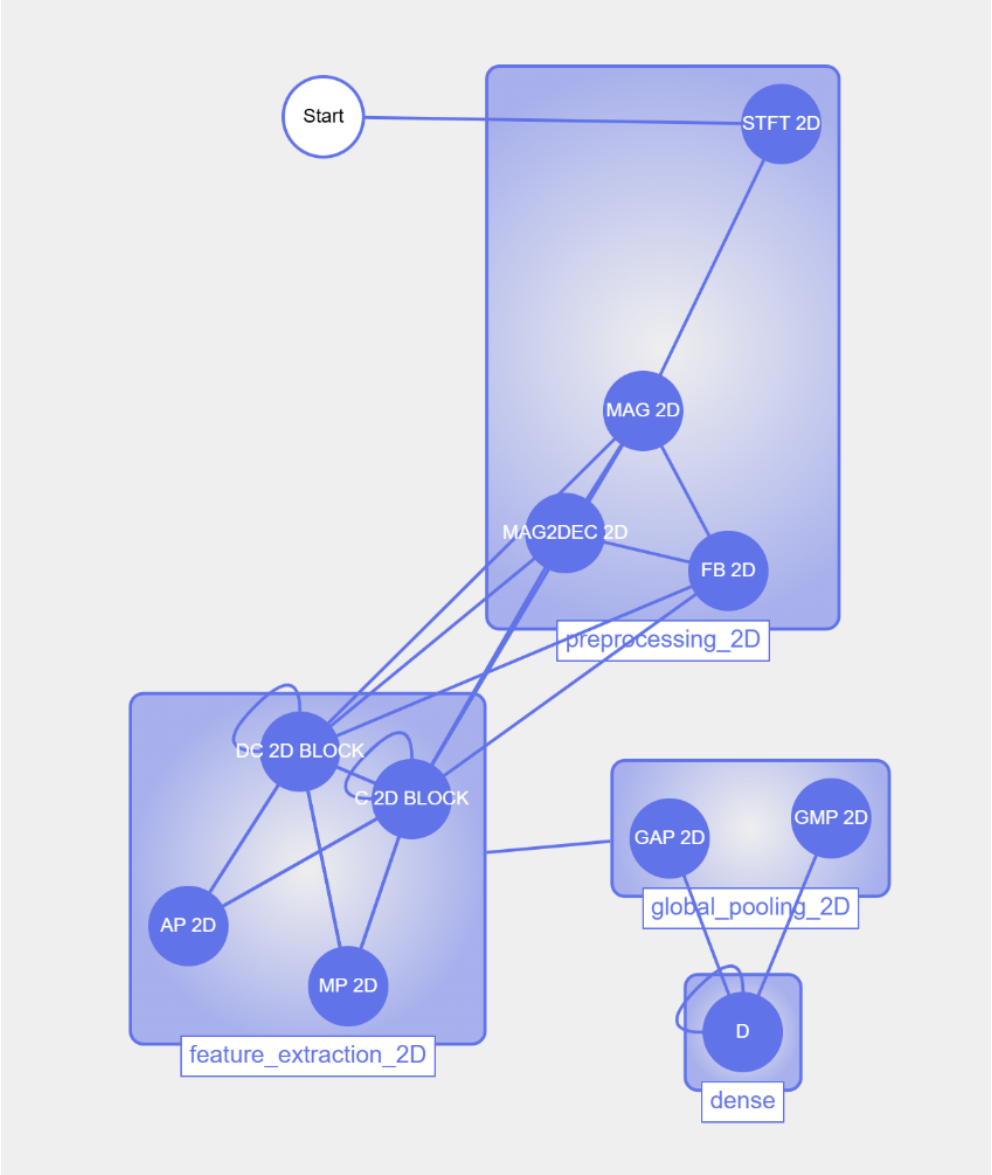

Fig. C1 Search space visualization of the EvoVis dashboard where the search space and the rule set are both used.

Appendix D Available operations in TFLite Micro/LiteRT

Table D1: List of TensorFlow Lite Micro operations with indication if the respective kernel is available in CMSIS-NN, which offers faster implementations.

Operation	Description	CMSIS-NN
Abs	Computes the absolute value of input tensor elements	×
Add	Element-wise addition of two tensors	✓
AddN	Adds all input tensors element-wise	×
ArgMax	Returns the indices of the maximum values along an axis	×
ArgMin	Returns the indices of the minimum values along an axis	×
AssignVariable	Assigns a value to a variable	×
AveragePool2D	Performs average pooling on the input	✓
BatchMatMul	Performs batch matrix multiplication	✓
BatchToSpaceNd	Reshapes batch dimension into spatial dimensions	×
BroadcastArgs	Returns the shape of broadcasted tensors	×
BroadcastTo	Broadcasts an array to a compatible shape	×
CallOnce	Calls a function once	×
Cast	Casts a tensor to a new type	×
Ceil	Computes ceiling of elements in the input tensor	×
CircularBuffer	Implements a circular buffer data structure	×
Concatenation	Concatenates tensors along one dimension	×
Conv2D	Performs 2D convolution	✓
Cos	Computes cosine of input elements	×
CumSum	Computes the cumulative sum of elements along an axis	×
Delay	Signal processing operation that introduces time delay	×
DepthToSpace	Rearranges depth data into spatial dimensions	×
DepthwiseConv2D	Performs depthwise 2D convolution	✓
Dequantize	Converts quantized tensors back to floating-point	×
DetectionPostprocess	Post-processing for object detection models	×

Continued on next page

Table D1 – Continued from previous page

Operation	**Description**	**CMSIS-NN**
Div	Element-wise division of two tensors	×
EmbeddingLookup	Looks up embeddings for specified IDs	×
Energy	Computes signal energy in signal processing	×
Elu	Exponential Linear Unit activation function	×
Equal	Element-wise equality comparison	×
EthosU	Operations for Arm Ethos-U NPU	×
Exp	Computes exponential of input elements	×
ExpandDims	Expands tensor dimensions by inserting a dimension	×
FftAutoScale	Auto-scales Fast Fourier Transform output	×
Fill	Creates a tensor filled with a scalar value	×
FilterBank	Applies filter bank in signal processing	×
FilterBankLog	Applies logarithmic filter bank	×
FilterBankSquareRoot	Applies square root to filter bank outputs	×
FilterBankSpectralSubtraction	Applies spectral subtraction to filter bank	×
Floor	Computes floor of elements in the input tensor	×
FloorDiv	Integer division with floor operation	×
FloorMod	Returns element-wise remainder of division	×
Framer	Frames a signal into overlapping windows	×
FullyConnected	Performs fully connected layer operations	✓
Gather	Gathers slices from params tensor	×
GatherNd	Gathers slices from params tensor at indices	×
Greater	Element-wise greater than comparison	×
GreaterEqual	Element-wise greater than or equal comparison	×
HardSwish	Hard version of swish activation function	×
If	Conditional execution of subgraphs	×
Irfft	Inverse real Fast Fourier Transform	×
L2Normalization	L2 normalization of input tensor	×
L2Pool2D	L2 pooling operation	×
LeakyRelu	Leaky version of ReLU activation function	×
Less	Element-wise less than comparison	×
LessEqual	Element-wise less than or equal comparison	×
Log	Computes natural logarithm of input elements	×
LogicalAnd	Element-wise logical AND operation	×
LogicalNot	Element-wise logical NOT operation	×

Continued on next page

Table D1 – Continued from previous page

Operation	**Description**	**CMSIS-NN**
LogicalOr	Element-wise logical OR operation	×
Logistic	Computes sigmoid activation function	×
LogSoftmax	Computes log softmax activation function	×
Maximum	Element-wise maximum of two tensors	✓
MaxPool2D	Performs max pooling on the input	✓
MirrorPad	Pads a tensor using mirroring	×
Mean	Computes mean of elements across dimensions	×
Minimum	Element-wise minimum of two tensors	✓
Mul	Element-wise multiplication of two tensors	✓
Neg	Negates the value of each element	×
NotEqual	Element-wise inequality comparison	×
OverlapAdd	Reconstructs signal from overlapped frames	×
Pack	Packs a list of tensors into a single tensor	×
Pad	Pads a tensor with a constant value	✓
PadV2	Extended version of Pad with more options	×
PCAN	Per-Channel Affine Normalization for audio	×
Prelu	Parametric Rectified Linear Unit activation	×
Quantize	Quantizes a tensor from floating-point to integer	×
ReadVariable	Reads the value of a variable	×
ReduceMax	Computes maximum of elements across dimensions	×
Relu	Rectified Linear Unit activation function	×
Relu6	ReLU capped at 6 activation function	×
Reshape	Changes the shape of a tensor	×
ResizeBilinear	Resizes images using bilinear interpolation	×
ResizeNearestNeighbor	Resizes images using nearest neighbor interpolation	×
Rfft	Real Fast Fourier Transform	×
Round	Rounds values to the nearest integer	×
Rsqrt	Computes reciprocal square root	×
SelectV2	Selects elements from two tensors based on condition	×
Shape	Returns shape of a tensor	×
Sin	Computes sine of input elements	×
Slice	Extracts a slice from a tensor	×
Softmax	Computes softmax activation function	✓

Continued on next page

Table D1 – Continued from previous page

Operation	**Description**	**CMSIS-NN**
SpaceToBatchNd	Reshapes spatial dimensions into batch dimension	×
SpaceToDepth	Rearranges spatial data into depth dimension	×
Split	Splits a tensor into sub-tensors along a dimension	×
SplitV	Splits a tensor into sub-tensors with specified sizes	×
Squeeze	Removes dimensions of size 1 from the tensor shape	×
Sqrt	Computes square root of input elements	×
Square	Computes square of input elements	×
SquaredDifference	Computes squared difference between two tensors	×
StridedSlice	Extracts a strided slice from a tensor	×
Stacker	Stacks frames for signal processing	×
Sub	Element-wise subtraction of two tensors	×
Sum	Computes sum of elements across dimensions	×
Svdf	Singular Value Decomposition Filter operation	✓
Tanh	Computes hyperbolic tangent of input elements	×
TransposeConv	Performs transposed 2D convolution	✓
Transpose	Permutes dimensions of a tensor	✓
Unpack	Unpacks a tensor into a list of tensors	×
UnidirectionalSequenceLSTM	Applies LSTM to an input sequence	✓
VarHandle	Creates a resource variable handle	×
While	Executes a subgraph repeatedly	×
Window	Applies window function to signal frames	×
ZerosLike	Creates a tensor of zeros with same shape as input	×

Appendix E Visualization of optimization decays

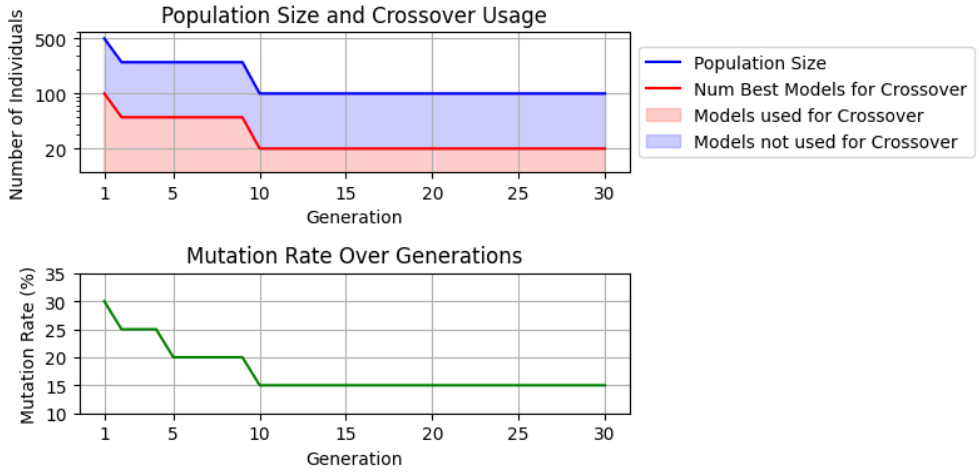

Fig. E1 Visualization of the decays used for population size and mutation rate. These are hyperparameter that are specified in the config file as well.

Appendix F Power consumption measurement

Fig. F1 Visualization of the current consumption measurement. The microcontroller is in an active/idle mode with a baseline current consumption visually around 4.6 mA. During inference (selected period), the average current consumption is 4.71 mA over 2.638 s, resulting in a measured charge of 12.42 mC. This data is thene used to calculate the energy consumption with the supply voltage of 3.3 V. Directly after inference ends, the microcontroller is set to sleep for 100 ms (indicated by the sharp drop in current).

Appendix G Statistical Testing of Mel-Scaling usage

Table G1 Shapiro-Wilk Test Results for Normality and Statistical Significance of Validation Accuracy

Normality (No n_mels)	**Normality (Has n_mels)**	**Significance (Val_Acc)**
	Worst 25%	
False ($p = 3.43 \times 10^{-37}$)	False ($p = 4.71 \times 10^{-22}$)	True ($p = 1.12 \times 10^{-23}$)
	Middle 50%	
False ($p = 5.68 \times 10^{-41}$)	False ($p = 2.14 \times 10^{-8}$)	True ($p = 2.24 \times 10^{-7}$)
	Best 25%	
False ($p = 2.72 \times 10^{-29}$)	True ($p = 0.216$)	True ($p = 0.00179$)