SONY

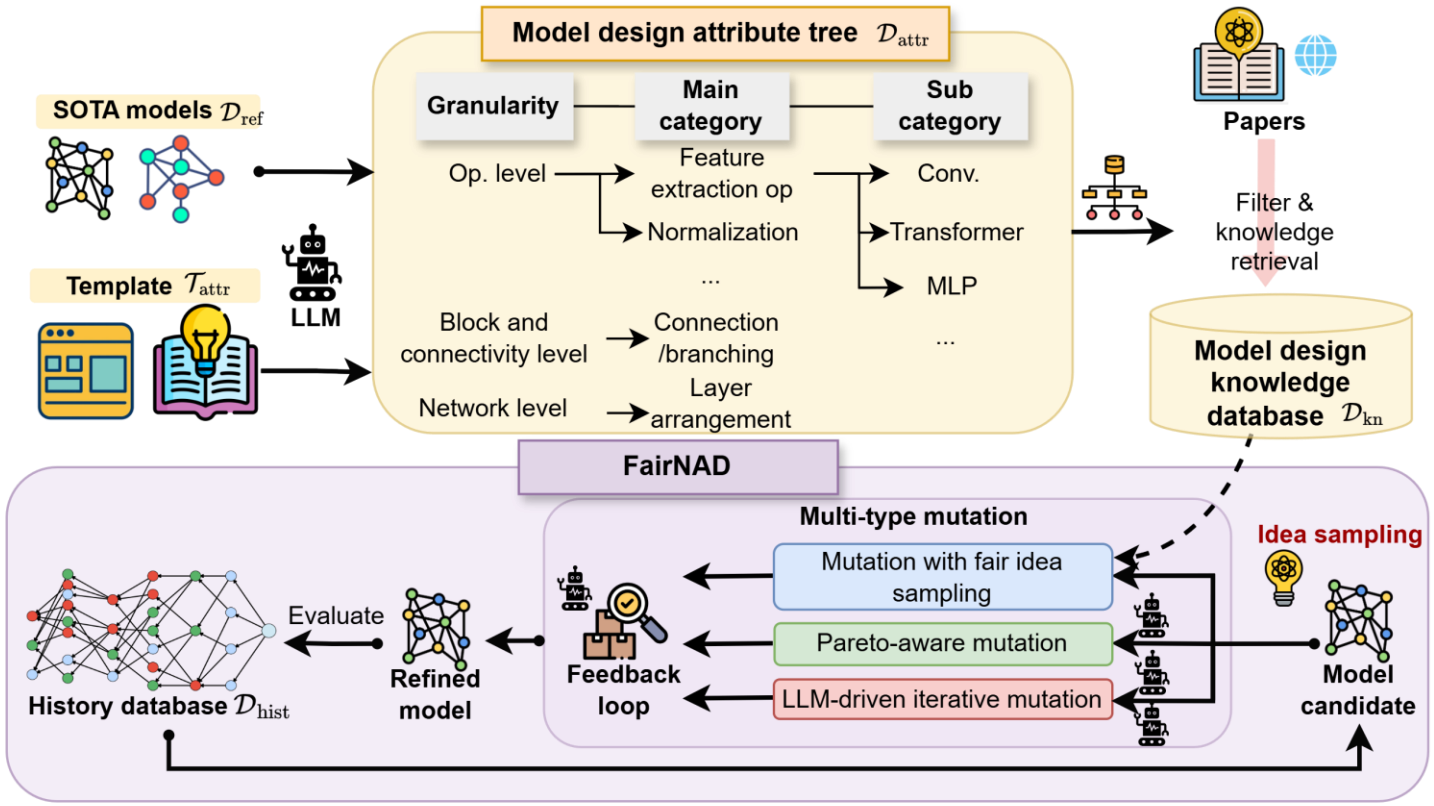
Structuring Open-Ended NAS: Semi-Automated Design Knowledge Structuring with LLMs for Efficient Neural Architecture Search

Yuiko Sakuma[1] · Masakazu Yoshimura[1] · Marcel Gröpl[2] · Zitang Sun[1] ·
Junji Otsuka[1] · Atsushi Irie[1] · Takeshi Ohashi[1]

[1] Sony Group Corporation [2] ETH Zürich

Overview

We propose FairNAD, an LLM-driven open-ended neural architecture search (NAS) that structures architectural design knowledge from research papers for efficient search



FairNAD achieves state-of-the-art results on NAS-Bench-201

CIFAR-10	**+0.84** pts
CIFAR-100	**+2.17** pts
ImageNet16-120	**+2.35** pts

Motivation

Conventional hand-crafted NAS

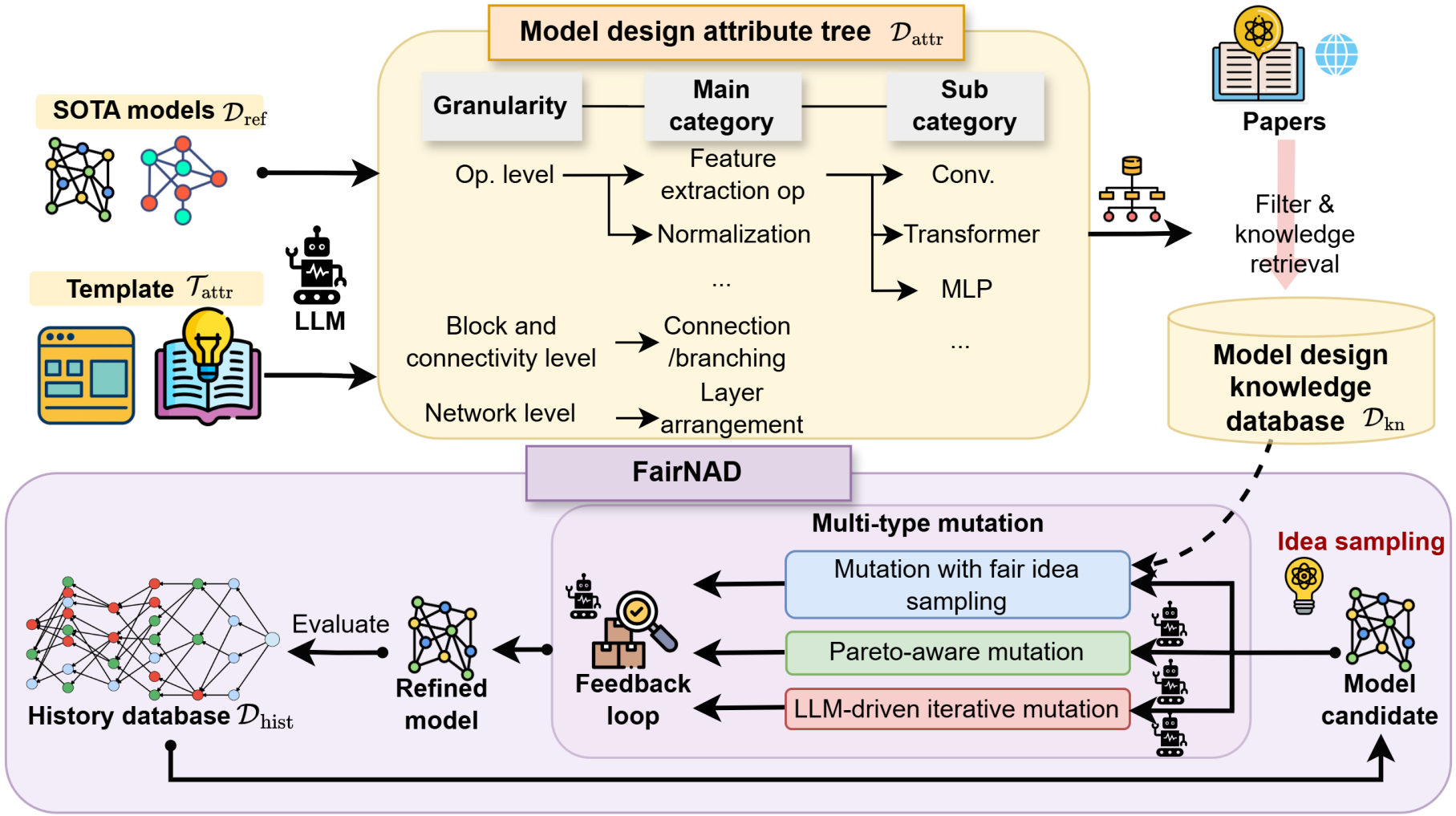
- Search space is constrained by the engineer's knowledge
- Manual design is costly

LLM-driven open-ended NAS

- Exploration beyond predefined search space is enabled
- Using design knowledge extracted from research papers often suffers from bias and quality issues

We propose a semi-automated approach that structures the open-ended search space

Semi-Automated Design Knowledge Structuring



- Model design knowledge is extracted from papers
- Each design idea is assigned to an attribute category

- Ideas from different attribute categories are uniformly sampled

Design Ideas of Different LLM-driven NAS Methods

Predefined

Limited diversity
- Add a layer
- Delete a layer
- Reduce the size of the network
- Add convolutional layer
- Add pooling layer
- Add residual connection
- ...

Open-ended / Unstructured

Not related to visual backbone design
- Incorporate gradient norm aware regularization directly into the loss function
- Design a custom decoder head to handle dense predictions

Task-specific ideas
- Utilize neural rendering via NeRF

Domain-specific ideas
- Use depth maps projection from 3D point clouds

Open-ended / Structured

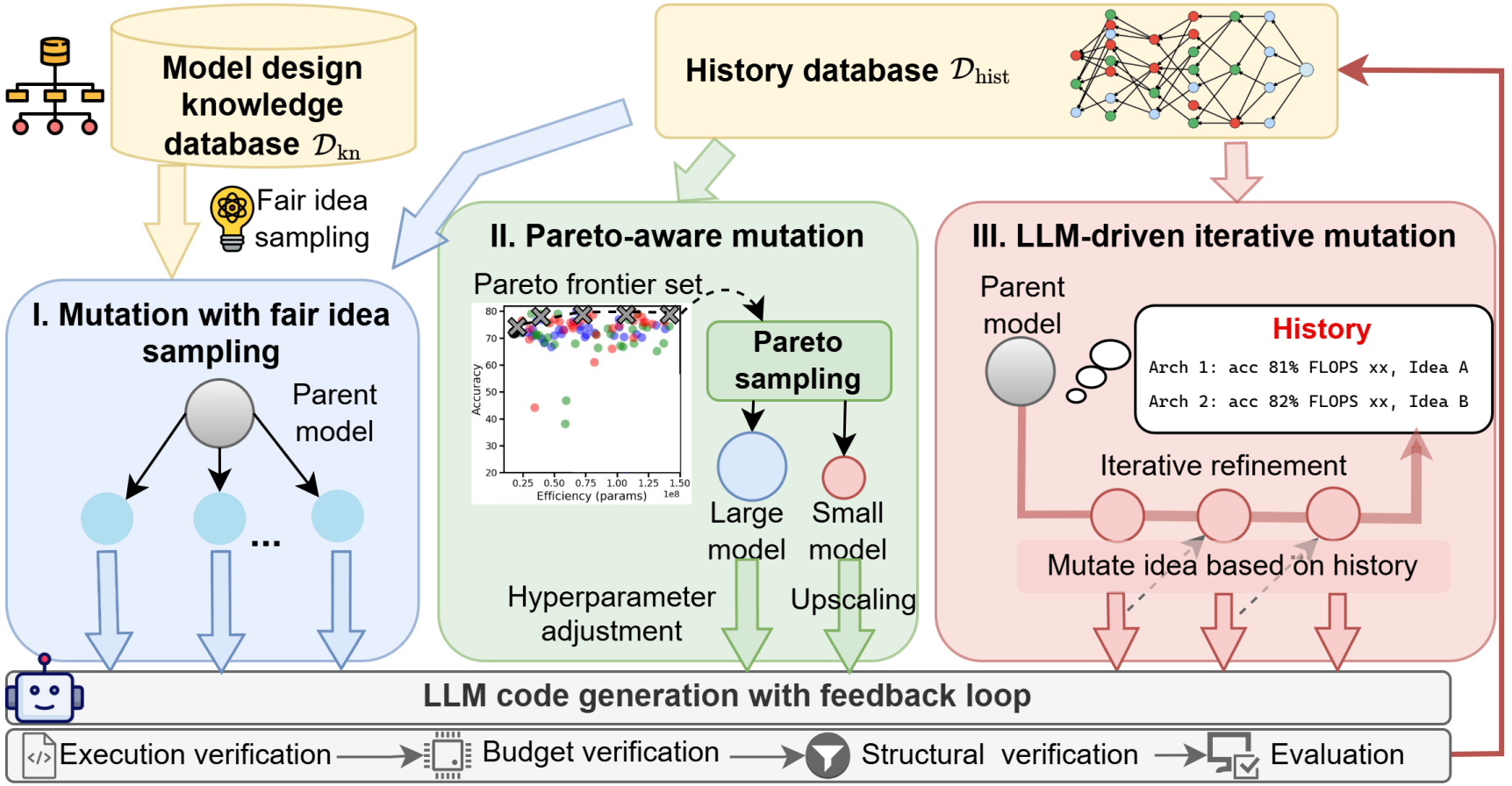
Filtered and fine-grained ideas

[Idea]
Use grouped convolution with dynamic filters for spatial feature extraction

[Tag]
Granularity: operation level
Main category: feature extraction op.
Sub-category: grouped convolution

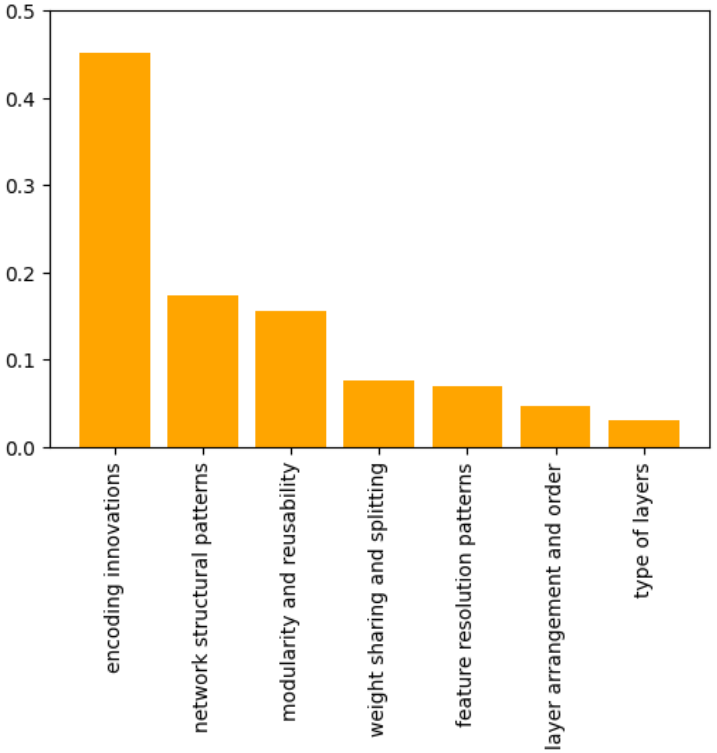
FairNAD

The FairNAD framework employs a multi-stage search process to efficiently and effectively explore the open-ended search space

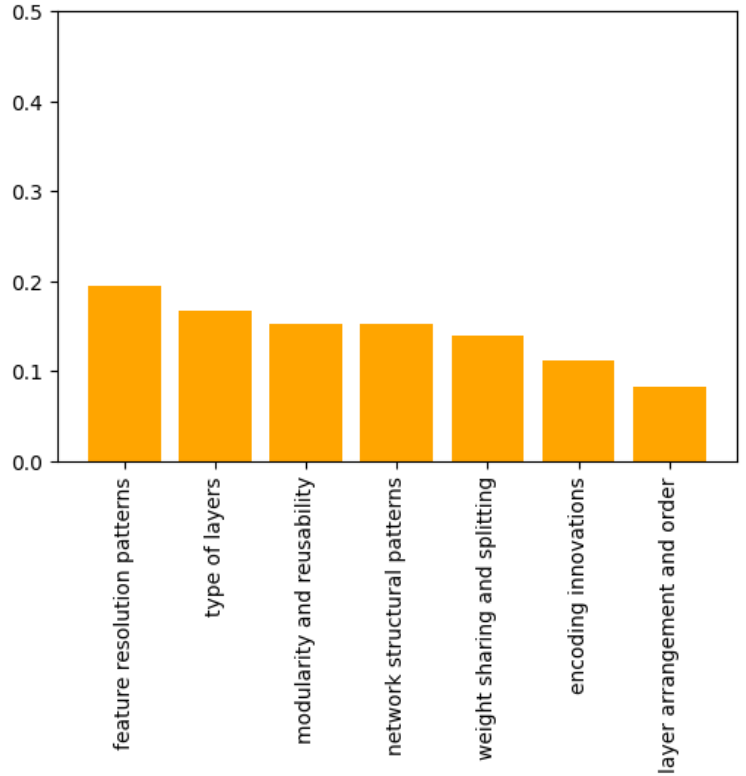


Effect of Fair Idea Sampling

Fair idea sampling helps to explore diverse design ideas



Design knowledge extracted from papers is biased



Fair idea sampling enables balanced search

Benchmarking on NAS-Bench-201

We evaluated FairNAD on

- CIFAR-10
- CIFAR-100
- ImageNet16-120

with following restriction:

< 1.5M params &

< 200M FLOPs

	Method	Search (archs)	CIFAR-10 validation	CIFAR-10 test	CIFAR-100 validation	CIFAR-100 test	ImageNet16-120 validation	ImageNet16-120 test
	ResNet [21] (base model)	-	90.83	93.97	70.42	70.86	44.53	43.63
	NAS Bench 201 Optimal	-	91.61	94.37	73.49	73.51	46.77	47.31
hand crafted search space	◇ ENAS [38]	-	37.51±3.19	53.89±0.58	13.37±2.35	13.96±2.33	15.06±1.95	14.84±2.10
	◇ DARTS [29]	-	39.77±0.00	54.30±0.00	38.57±0.00	15.61±0.00	18.87±0.00	16.32±0.00
	◇ SETN [15]	-	84.04±0.28	87.64±0.00	58.86±0.06	59.05±0.24	33.06±0.02	32.52±0.21
	◇ DSNAS [23]	-	89.66±0.29	93.08±0.13	30.87±16.40	31.01±16.38	40.61±0.09	41.07±0.09
	◇ PC-DARTS [54]	-	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22
	◇ SNAS [53]	-	90.10±1.04	92.77±0.83	69.69±2.39	69.34±1.98	42.84±1.79	43.16±2.64
	◇ iDARTS [59]	-	89.86±0.60	93.58±0.32	70.57±0.24	70.83±0.48	40.38±0.59	40.89±0.68
	◇ GDAS [16]	-	89.89±0.08	93.61±0.09	71.34±0.04	70.70±0.30	41.59±1.33	41.71±0.98
	◇ DRNAS [9]	-	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
	◇ β-DARTS [57]	-	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
	◇ A-DARTS [36]	-	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
	♡ REA [40]	500	91.19±0.31	93.92±0.30	71.81±1.12	71.84±0.99	45.15±0.89	45.54±1.03
	♡ RS [2]	500	90.93±0.36	93.70±0.36	70.93±1.09	71.04±1.07	44.45±1.10	44.57±1.25
	♡ REINFORCE [49]	500	91.09±0.37	93.85±0.37	71.61±1.12	71.71±1.09	45.05±1.02	45.24±1.18
	♡ BOHB [19]	500	90.82±0.53	93.61±0.52	70.74±1.29	70.85±1.28	44.26±1.36	44.42±1.49
	★ GENIUS [60]	10	91.07±0.20	93.79±0.09	70.96±0.33	70.91±0.72	45.29±0.81	44.96±1.02
	★ CoLLM-NAS [27]	100	91.59±0.04	94.37±0.01	73.44±0.12	73.44±0.15	46.62±0.10	46.79±0.28
open-ended NAS	★ LLMatic [37]	2000	-	94.26±0.13	-	71.62±1.73	-	45.87±0.96
	★ LeMo-NADe(Gemini) [39]	30	82.94	81.76	52.12	52.96	30.34	31.02
	★ LeMo-NADe(GPT4) [39]	30	90.90	89.41	63.38	67.90	27.05	27.70
	★ NADER [56]	500	91.55	94.62	75.72	76.00	50.20	50.52
	★ FairNAD	100	91.77±0.29	94.93±0.17	76.54±0.28	76.26±0.33	49.99±0.80	49.75±0.61
		500	**92.20±0.14**	**95.46±0.10**	**78.75±0.99**	**78.17±1.25**	**52.73±1.74**	**52.87±1.32**

+0.84 point **+2.17** point **+2.35** point

Example of the Discovered Architectures

```python
class BaseBlock(nn.Module):
    def __init__(self, in_channels, out_channels, stride=1):
        super().__init__()
        self.local_conv_attention = LocalConvAttention(in_channels,
     out_channels, stride=stride)
        self.op4 = nn.Conv2d(out_channels, out_channels,
    kernel_size=1, stride=1, padding=0, bias=False)
        self.op5 = nn.BatchNorm2d(out_channels)
        self.downsample = (in_channels != out_channels) or (stride
    != 1)
        if self.downsample:
            self.proj = nn.Conv2d(in_channels, out_channels,
    kernel_size=1, stride=stride, padding=0, bias=False)
            self.bn = nn.BatchNorm2d(out_channels)
        self.op6 = nn.ReLU(inplace=True)

    def forward(self, x):
        x1 = self.local_conv_attention(x)
        x_concat = x1
        x_concat = self.op4(x_concat)
        x_concat = self.op5(x_concat)
        if self.downsample:
            x = self.proj(x)
            x = self.bn(x)
        x_concat = x_concat + x
        x_concat = self.op6(x_concat)
        return x_concat
```

Incorporates innovative modules

```python
class Network(nn.Module):
    def __init__(self, num_classes=120):
        super().__init__()
        self.stem = nn.Sequential(
            nn.Conv2d(3, 51, kernel_size=3, padding=1, bias=False),
            PositionalEncoding(51),
            nn.BatchNorm2d(51),
            nn.ReLU(inplace=True)
        )
        layers = []
        for _ in range(8):
            layers.append(BaseBlock(51, 51, stride=1))
        self.layers1 = nn.Sequential(*layers)
        layers = []
        layers.append(BaseBlock(51, 102, stride=2))
        for _ in range(15):
            layers.append(BaseBlock(102, 102))
        self.layers2 = nn.Sequential(*layers)
        layers = []
        layers.append(BaseBlock(102, 204, stride=2))
        layers.append(BaseBlock(204, 204, stride=2))  # Added second
    downsampling block
        self.layers3 = nn.Sequential(*layers)
        self.avg_pool = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(204, num_classes)

    def forward(self, x):
        h = self.stem(x)
        h = self.layers1(h)
        h = self.layers2(h)
        h = self.layers3(h)
        h = self.avg_pool(h)
        h = h.view(h.size(0), -1)
        h = self.fc(h)
        return h
```

Introduces diverse components used both in CNNs and Transformers

Provides flexible control over network width and depth

Conclusions

We proposed an effective and efficient open-ended NAS method:

- Semi-automated model design knowledge structuring extracts diverse and curated knowledge
- FairNAD uses multi-type mutations that balance exploration and exploitation by uniformly exploring ideas from different category and focusing on high-performing models

Future works include:

- Open-set extension
- Base LLM fine-tuning

SONY