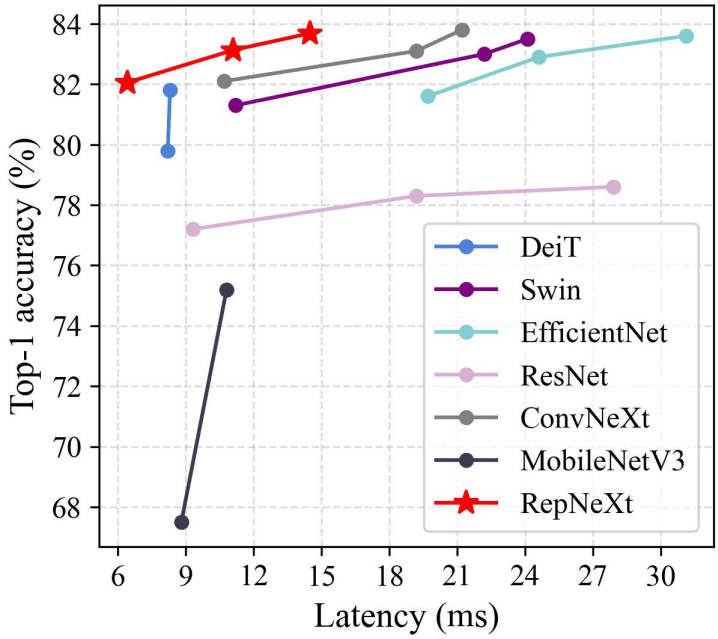
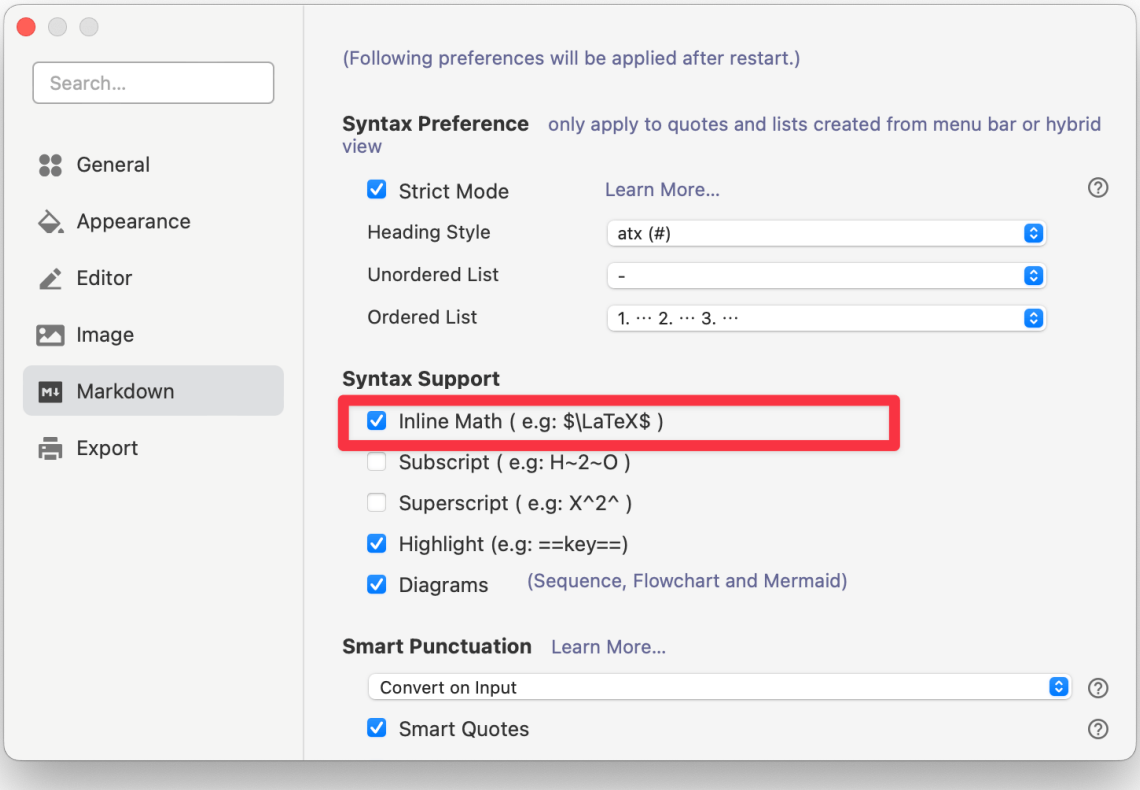
RepNeXt: A Re-parameterizable CNN with High Performance and Efficiency



- We propose a re-parameterizable ConvNet named RepNeXt, which may be a riveting attempt to achieve both high performance and efficiency.
- We design a multi-scale re-parameterizable structure as the main feature extractor. It is simple and easy to implement without excessively increasing the training cost and proved to be highly applicable to fine-grained and low-resolution tasks.
- We devise a general shortcut re-parameterization to eliminate the shortcut branches, which can bring a substantial acceleration to the inference of a network by improving its memory access and parallelism.

```
RepNeXt
├─ README.md
├─ README.pdf
├─ An_Introduction_to_Structural_Reparameterization.md
├─ An_Introduction_to_Structural_Reparameterization.pdf
├─ Training.md
├─ Training.pdf
├─ rep_validation.py
├─ reparameterizer.py
├─ repnext.py
└─ repunit1and2.py
```

`An_Introduction_to_Structural_Reparameterization.md`: A detailed introduction to structural re-parameterization, including the meaning, paradigms, derivations, and codes. To display the formula properly, please turn on the formula inline function of your markdown reader. Take `typora` as an example (The **PDF** files with the same content as the **markdown** files are prepared for convenient navigation):



`reparameterizer.py`: A tool class for implementing different kinds of structural re-parameterization operations. This class inherits from `nn.Module` and it will be convenient to inherit this class to implement various re-parameterizable models.

`Training.md`: A description for training RepNeXt on datasets.

`repnext.py`: Definition of the RepNeXt's model architecture (`RepNeXt_u3`).

`repunit1and2.py`: `repnext.py` implements `RepNeXt_u3`. When you need to use `RepNeXt_u1` and `RepNeXt_u2`, you can use the code snippet in this file to quickly rebuild the network structure.

`rep_validation.py`: It is used to quickly appreciate the speed difference and the consistency of calculation results before and after structural re-parameterization. The usage is shown below.

```
python3 rep_validation.py --operation consistency_comparation
# or
python3 rep_validation.py --operation speed_comparation
```

> Important libraries that may be needed: `numpy`, `timm`, `torch`, `torchvision`, `tqdm`, `pillow`