# EZNAS: Evolving Zero-Cost Proxies For Neural Architecture Scoring

Yash Akhauri<sup>1</sup> J. Pablo Muñoz<sup>2</sup> Nilesh Jain<sup>2</sup> Ravi Iyer<sup>2</sup> <sup>1</sup>Cornell University <sup>2</sup>Intel Labs ya255@cornell.edu {pablo.munoz,nilesh.jain,ravishanker.iyer}@intel.com

# 1 Appendix

#### 1.1 NASBench-201 and NDS

For image classification, we utilize the NASBench-201 Dong and Yang [2020] and NDS Radosavovic et al. [2019] NAS search spaces for our evolutionary search as well as testing. NASBench-201 consists of 15,625 neural networks trained on the CIFAR-10, CIFAR-100 and ImageNet-16-120 datasets. Neural Networks in Network Design Spaces (NDS) uses the DARTS Liu et al. [2019] skeleton. The networks are comprised of cells sampled from each of AmoebaNet Real et al. [2019], DARTS Liu et al. [2019], ENAS Pham et al. [2018], NASNet Zoph et al. [2018] and PNAS Liu et al. [2018]. There exists approximately 5000 neural network architectures in each NDS design space.

#### 1.2 Sequential Program Representation

Our initial attempts at discovering ZC-NASMs took a different approach to program representation. The sequential program representation described in Figure 1 posed no structural limitations on the program. We have 22 static memory addresses, which contained network statistics and are referenced with integers 0-21. To store intermediate tensors generated by the program, we allocate 80 dynamic memory addresses, which can be over-written during the program execution as well. To store intermediate scalars generated by the program, we allocate 20 memory addresses. As seen in Figure 1, we represent the programs as integers, where each instruction is expressed as 4 integers. The first integer provides the write address, the second integer provides the operation



Figure 1: Our sequential program representation.

ID and the third and fourth integers provide the read addresses for the operation. We initialize *valid* random integer arrays and convert them to programs to evaluate and fetch the fitness (score-accuracy correlation). We allow Mate, InsertOP, RemoveOP, MutateOP & MutateOpArg as variation functions. The Mate function takes two individuals, and takes the first half of each individual. Then, these components are interpolated to generate a new individual. The InsertOP function inserts an operation at a random point in the program. The RemoveOP function removes an operation at a random point in the program. The RemoveOP function in program without changing read/write addresses. The MutateOpArg function simply replaces one of the read arguments of any random instruction with another argument from the same address space (dynamic address argument cannot be replaced by a scalar address argument).

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



Figure 2: (Left) Experiment demonstrating the effect of seed on our score-accuracy correlation for neural network initialization with a batch size of 1. (Right) Effect of seed on our score-accuracy correlation with a batch size of 1. CIFAR and ImageNet abbreviated as CF and IN respectively. This test was done on each design space over 7 seeds for 400 Neural Networks.

While we are able to discover weak ZC-NASMs with this formulation, we observe that there are too many redundancies in the programs discovered. Program length bloating as well as operations that do not contribute to the final output were frequently observed. Due to these issues, the evolution time evaluation of individual fitness quickly became an intractable problem. To address this, we change our program representation to a expression tree representation in the results reported in the paper. This representation necessitates contribution of each operation to the final output, which means there is no redundant compute. While the sequential program representation is valid, we believe that significant engineering efforts are required to ensure discovery of meaningful programs. Our sequential program representation is directly inspired by the formulation used in AutoML-Zero Real et al. [2020]. AutoML-Zero makes significant approximations in the learning task to evolutionarily discover MLPs. While AutoML-Zero has a much larger program space to search for, approximations in computing individual fitness are not feasible in our formulation as generating exact score-accuracy correlation is an important factor in selecting individuals with high fitness.

#### 1.3 Noise and Perturbation for Network Statistics

To generate network statistics, we use three types of input data. The first is simply a single random sample from the dataset (e.g. a single image or a batch of images from CIFAR-10). To generate a noisy input, we simply use the default torch.randn function as input = torch.randn(data\_sample.shape). The third type of input we provide is a data-sample which has been perturbed by random noise (input = data\_sample +  $0.01**0.5*torch.randn(data_sample.shape)$ ).

#### 1.4 Network Initialization Seed Test

In Figure 2 (Left), we use different seeds to change the initialization and input tensors, but keep the neural architectures being sampled fixed in the respective spaces. The variance in the score accuracy correlation is much lesser than in Figure 2 (Right) where the seed also controls the neural architectures being sampled. This shows the true variation in our EZNAS-A ZC-NASM with respect to network initialization.



Figure 3: Experiment demonstrating the ability to discover ZC-NASMs with an alternate network statistics collection strategy and to\_scalar function. Experiments are named as EZNAS-(Statistics Collection Structure)-(to\_scalar). Two statistics collection structures are tested. (R-C-B) is a ReLU-Conv2D-BatchNorm2D structure, (C-B-R) is a Conv2D-BatchNorm2D-ReLU structure. (to\_scalar) can be Mean or L2.

Kendall Tau	NASBench201 CIFAR10	NASBench201 CIFAR100	NASBench201 ImageNet16-120	Amoeba CIFAR10	DARTS CIFAR10	ENAS CIFAR10	PNAS CIFAR10	NASNet CIFAR10	FLOPs	Params
NASBench201 CIFAR10	0.533	-0.499	0.4345	0.4422	<u>0.5777</u>	0.472	0.4514	0.2633	<u>0.5610</u>	0.5610
NASBench201 CIFAR100	0.5242	-0.496	0.4522	0.4152	<u>0.5627</u>	0.476	0.453	0.2531	<u>0.5472</u>	0.5472
NASBench201 ImageNet16-120	0.4575	-0.424	0.4350	0.4213	<u>0.5733</u>	0.456	0.2903	0.2338	<u>0.5036</u>	0.5036
Amoeba CIFAR10	0.1146	-0.029	0.3263	0.352	<u>0.3774</u>	0.380	0.35278	<u>0.4042</u>	0.2614	0.2658
DARTS CIFAR10	0.3236	-0.173	0.2088	0.4187	<u>0.5077</u>	0.460	0.47570	0.1439	<u>0.5079</u>	0.5042
ENAS CIFAR10	0.2833	-0.143	0.3132	0.4292	0.3795	0.422	0.41701	0.3400	<u>0.4739</u>	<u>0.4704</u>
PNAS CIFAR10	0.2208	-0.053	0.4212	0.4466	0.4435	<u>0.514</u>	<u>0.4874</u>	0.3799	0.3363	0.3223
NASNet CIFAR10	0.2370	0.0229	0.2880	<u>0.3789</u>	<u>0.4381</u>	0.364	0.3594	0.3757	0.1996	0.2102

Figure 4: Full correlation table. Each column represents the dataset evolution was performed on. The DARTS-CIFAR10 column is the EZNAS-A NASM. Each row represents the dataset the best discovered NASM program was tested on. Best score-accuracy KTR in bold and underlined. Second best score-accuracy KTR in italics and underlined. These tests are done by evolving on 100 neural networks and testing on the test task dataset (1000 randomly sampled neural networks on NASBench-201 and 200 randomly sampled neural networks on NDS). The network statistics were generated with a batch size of 1.

# 1.5 Hardware used for evolution and testing

Our evolutionary algorithm runs on Intel(R) Xeon(R) Gold 6242 CPU with 630GB of RAM. Our RAM utilization for evolving programs on a single Image Classification dataset was approximately 60GB. RAM utilization can vastly vary (linearly) based on the number of neural network statistics that are being used for the evolutionary search. Our testing to generate the statistics for the seed experiments as well as the final Spearman  $\rho$  and Kendall Tau Rank Correlation Coefficient is done on an NVIDIA DGX-2 server with 4 NVIDIA V-100 32GB GPUs.

# 1.6 Varying network statistics and to\_scalar

In Figure 3, we detail 3 tests while evolving on the NDS\_DARTS CIFAR-10 search space in an identical fashion to EZNAS-A (referred to as EZNAS-(R-C-B)-(Mean) here). EZNAS-(C-B-R)-(Mean) and EZNAS-(C-B-R)-(L2) correspond to alternate to\_scalar and network statistics collection tests respectively. We demonstrate that while EZNAS-(R-C-B)-(Mean) is more effective, we are able to discover ZC-NASMs with all three formulations.

# 1.7 Hyper-parameters for discovering EZNAS-A

```
num_generation: 15
population_size: 50
tournament_size: 4
MU: 25
lambda_: 50
crossover_prob: 0.4
mutation_prob: 0.4
min_tree_depth: 2
max_tree_depth: 10
```

# References

- [1] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search, 2020.
- [2] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search, 2018.
- [3] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019.
- [4] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing, 2018.
- [5] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition, 2019.
- [6] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search, 2019.
- [7] Esteban Real, Chen Liang, David R. So, and Quoc V. Le. Automl-zero: Evolving machine learning algorithms from scratch, 2020.
- [8] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2018.

OP0Element-wise Sum $C = A+B$ OP1Element-wise Product $C = A+B$ OP3Matrix Multiplication $C = A+B$ OP4Lesser Than $C = (A+B)$ .bool()OP5Greater Than $C = (A+B)$ .bool()OP6Equal To $C = (A+B)$ .bool()OP7Log $C = torch.log(A)$ Af[A=0] = 1 $C = torch.log(A)$ OP8AbsLog $C = torch.log(torch.abs(A))$ OP9Abs $C = torch.log(torch.abs(A))$ OP10Power $C = torch.log(torch.abs(A))$ OP11Exp $C = torch.log(torch.abs(A))$ OP12Normalize $C = (T-C), pov(A, 2)$ OP13ReLU $C = torch.log(torch.abs(A))$ OP14Sign $C = torch.log(torch.abs(A))$ OP15Heaviside $C = torch.functional.F.relu(A)$ OP16Element-wise Invert $C = 1/A$ OP17Frobenius Norm $C = torch.log(det(A))$ OP18Determinant $C = torch.log(A)$ OP20SymEigRatio $A = A + A.T$ $e = (-1)/e[0]$ $A = Ae(A.T)$ $A = Ae(A.T)$ $e = (-1)/e[0]$ $A = Arceh.T$ $A = Arceh.T) (A, A)$ OP21EigRatio $C = torch.sum(A)/A.numel()$ OP22Normalized Sum $C = torch.sum(A)/A.numel()$ OP23L1 Norm $A = Areahape(A.shape[0], -1)$ $A = Areahape(A.shape[0], -1)$ $A = Areahape(A.shape[0], -1)$ $A = Reaviside(A)$ $C = (-1)/e[0]$ OP24Hamming Distance $B = Heaviside(A)$ $OP25$ K	Op ID	Operation	<b>Description</b> Output: C, Input: A, B					
OP1Element-wise Difference $C = A-B$ OP2Element-wise Product $C = A+B$ OP3Matrix Multiplication $C = A+B$ OP4Lesser Than $C = (A+B).bool()$ OP5Greater Than $C = (A+B).bool()$ OP6Equal To $C = (A+B).bool()$ OP7Log $C = torch.log(A)$ AfA $A[A=0] = 1$ OP8AbsLog $C = torch.log(b)$ OP9Abs $C = torch.log(b)$ OP10Power $C = torch.log(b)$ OP11Exp $C = torch.log(b)$ OP12Normalize $C[C](-C] = 0$ OP13ReLU $C = torch.nexp(A)$ OP14Sign $C = torch.norm(A, p^{-1}relu(A)$ OP15Heaviside $C = torch.norm(A, p^{-1}ro^{-1})$ OP16Element-wise Invert $C = torch.norm(A, p^{-1}ro^{-1})$ OP17Frobenius Norm $C = torch.norm(A, p^{-1}ro^{-1})$ OP18Determinant $C = torch.spmej(A)$ OP19LogDeterminant $C[c](-C]=0$ $A = A + A.T$ $e, v = torch.sig(A)$ OP20SymEigRatio $A = A + A.T$ $e, v = torch.sig(A)$ $C = torch.sig(A)$ OP21Normalized Sum $C = torch.sig(A).fill(A)$ OP22Normalized Sum $C = torch.sig(A).fill(A)$ OP23L1 Norm $A = A + A.T$ $e, v = torch.sig(A).fill(A)$ $e = sum(A) + B)$ OP24Hamming Distance $B = Heaviside(A)$ OP25KL Divergence $C = torch.nores(Lashape(I), -1)$ $A = A - eshape(A.shap$	OP0	Element-wise Sum	C = A + B					
OP2Element-wise Product $C = A*B$ OP3Matrix Multiplication $C = A*B$ OP4Lesser Than $C = (A*B).bool()$ OP5Greater Than $C = (A*B).bool()$ OP6Equal To $C = (A*B).bool()$ OP7Log $C = (A*B).bool()$ OP8AbsLog $C = (A*B).bool()$ OP9Abs $C = torch.log(A)$ OP10Power $C = torch.opx(A, 2)$ OP11Exp $C = torch.opx(A, 2)$ OP12Normalize $C = torch.opx(A, 2)$ OP13ReLU $C = torch.sign(A)$ OP14Sign $C = torch.sign(A)$ OP15Heaviside $C = torch.orm(A, p='fro')$ OP16Element-wise Invert $C = 1/A$ OP17Frobenius Norm $C = torch.logdet(A)$ OP18Determinant $C = torch.sign(A)$ OP19LogDeterminant $C = torch.sign(A)$ OP20SymEigRatio $A = A + A.T$ $e, v = torch.eig(A)$ $C = torch.sign(A) < numel()$ OP21EigRatio $A = torch.eis(A) < numel()$ OP22Normalized Sum $C = torch.sign(A) < numel()$ OP24Hamming Distance $B = Heaviside(B)$ OP25KL Divergence $C = torch.nume(A) < numel()$ OP26Cosine Similarity $B = Reshape(A.shape[0], -1)$ OP27Softmax $C = torch.sine(A) < numel()$ OP28Sigmoid $C = torch.nume(A) < numel(A)$ OP24Hamming Distance $B = Heaviside(A)$ OP25KL Divergence $C = torch.sine$	OP1	Element-wise Difference	C = A - B					
OP3Matrix Multiplication $C = AGB$ OP4Lesser Than $C = (A \times B).bool()$ OP5Greater Than $C = (A \times B).bool()$ OP6Equal To $C = (A \times B).bool()$ OP7Log $C = torch.log(A)$ Alk<=0] = 1	OP2	Element-wise Product	C = A * B					
OP4Lesser Than $C = (A < B) . bool ()$ $OP5$ Greater Than $C = (A < B) . bool ()$ $OP6$ Equal To $C = (A < B) . bool ()$ $OP7$ Log $A[A < O] = 1$ $OP7$ Log $C = torch . log(A)$ $OP8$ AbsLog $A[A = 0] = 1$ $OP9$ Abs $C = torch . log(Corch . abs(A))$ $OP10$ Power $C = torch . log(Corch . abs(A))$ $OP10$ Power $C = torch . log(A / A / A / A / A / A / A / A / A / A /$	OP3	Matrix Multiplication	C = AOB					
OP5 OP6 Equal To $C = (A>B).bool()$ $C = (A=B).bool()$ OP7Log $A[A=0] = 1$ 	OP4	Lesser Than	$C = (A \leq B), bool()$					
OP6Equal To $C = (A==B), bool()$ OP7Log $A[A<=0] = 1$ OP8AbsLog $A[A=0] = 1$ OP9Abs $C = torch.log(A)$ OP10Power $C = torch.abs(A)$ OP11Exp $C = torch.abs(A)$ OP12Normalize $C = torch.abs(A)$ OP13ReLU $C = torch.functional.F.relu(A)$ OP14Sign $C = torch.functional.F.relu(A)$ OP15Heaviside $C = torch.abs(A)$ OP16Element-wise Invert $C = 1/A$ OP17Frobenius Norm $C = torch.norm(A, p='fro')$ OP18Determinant $C = torch.abs(A)$ OP19LogDeterminant $C = torch.abs(A)$ OP20SymEigRatio $A = A + A.T$ $e, v = torch.sum(A)(A.numel())$ $A = torch.sum(A)(A.numel())$ OP21EigRatio $A = torch.sum(A)(A.numel())$ OP24Hamming Distance $B = Heaviside(B)$ OP25KL Divergence $C = torch.nn.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ $A = A.reshape(A.shape[0], -1)$ $A = A.reshape(A.shape[0], -1)$ $A = A.reshape(A.shape[0], -1)$ OP24Hamming Distance $B = Heaviside(B)$ OP25KL Divergence $C = torch.nn.cosinlarity()(A, B)$ OP26Cosine Similarity $C = torch.functional.F.sigmoid(A)$ OP27Softmax $C = torch.nes_like(A)$ OP29Ones Like $C = torch.ares_like(A)$ OP20Sigmoid $C = torch.ares_like(A)$ OP21EigRatio $C = torch.funct$	OP5	Greater Than	$C = (A > B) \cdot bool()$					
OP7Log $A[A<=0] = 1$ OP8AbsLog $C = torch.log(A)$ OP9Abs $C = torch.log(torch.abs(A))$ OP9Abs $C = torch.log(torch.abs(A))$ OP10Power $C = torch.log(torch.abs(A))$ OP11Exp $C = torch.log(A)$ OP12Normalize $C[C!=C] = 0$ OP13ReLU $C = torch.exp(A)$ OP14Sign $C = torch.exign(A)$ OP15Heaviside $C = torch.neur(A)$ OP16Element-wise Invert $C = torch.neur(A)$ OP17Frobenius Norm $C = torch.logdet(A)$ OP18Determinant $C = torch.logdet(A)$ OP19LogDeterminant $C[C!=C] = 0$ OP20SymEigRatio $A = A + A.T$ $e, v = torch.sum(A)(A.nume1())$ $A = A + A.T$ $e, v = torch.sum(A)(A.nume1())$ $A = A + A.T$ $e, v = torch.sum(A)(A.nume1())$ $C = torch.sum(A)(A.nume1())$ OP21EigRatio $C = torch.sum(A)(A.nume1())$ OP23L1 Norm $C = torch.sum(A)(A.nume1())$ OP24Hamming Distance $B = Heaviside(B)$ OP25KL Divergence $C = torch.sum(C)$ OP26Cosine Similarity $C = torch.sum(C)$ OP27Softmax $C = torch.sum(C)$ OP28Sigmoid $C = torch.sum(C)$ OP29Ones Like $C = torch.sum(A)$ OP20Softmax $C = torch.sum(C)$ OP21EigRatio $C = torch.sum(A)$ OP23L1 Norm $C = torch.sum(A)$ OP24Hamming Dis	OP6	Equal To	C = (A == B), bool()					
OP7LogC = torch.log(A) $A[A=0] = 1$ (C = torch.log(torch.abs(A))OP8AbsLogC = torch.log(torch.abs(A))OP9AbsC = torch.log(torch.abs(A))OP10PowerC = torch.abs(A)OP11ExpC = torch.exp(A)OP12Normalize $C[C!=C] = 0$ OP13ReLUC = torch.functional.F.relu(A)OP14SignC = torch.sign(A)OP15HeavisideC = torch.norm(A, p='fro')OP16Element-wise InvertC = 1/AOP17Frobenius NormC = torch.abdt(A)OP18DeterminantC = torch.abdt(A)OP19LogDeterminantC = torch.abdt(A)OP20SymEigRatioA = A + A.TOP21EigRatioA = torch.einsum('nc,mc->nm', [A,A])OP22Normalized SumC = torch.sum(A)/A.numel()OP23L1 NormA = Heaviside(B)OP24Hamming DistanceB = Heaviside(B)OP25KL DivergenceC = torch.sum(A)/A.numel()OP26Cosine SimilarityC = torch.sum(A)/A.numel(), -1)OP27SoftmaxC = torch.sum(A)/A.stape[0], -1)OP24Hamming DistanceB = Heaviside(B)OP27SoftmaxC = torch.sum(A)/A.stape[0], -1)OP28KL DivergenceC = torch.sum(A)/A.stape[0], -1)OP29One LikeC = torch.sum(C)OP27SoftmaxC = torch.sum(A)OP29One LikeC = torch.sum(A)OP30Zeros LikeC = torch.sum(A)OP31 <td< td=""><td></td><td></td><td>A[A&lt;=0] = 1</td></td<>			A[A<=0] = 1					
OP8AbsLog $A[A=0] = 1$ ( $z = torch.log(trch.abs(A))$ ( $z = torch.log(A, 2)$ ( $DP10$ Power $C = torch.abs(A)$ ( $z = torch.abs(A)$ ( $C = torch.exp(A)$ ( $A = tausside$ ( $C = torch.exp(A)$ ( $C = torch.exp(A)$ ( $A = tausside$ ( $C = torch.exp(A)$ ( $A = tausside(A)$ ( $C = torch.exp(C)$ ( $C = torch.exp(A)$ ( $A = taussid(A)$ ( $A = taussid(A)$ ( $A = taussid(A)$ ( $C = torch.exp(A)$ ( $A = taussid(A)$ 	OP7	Log	C = torch.log(A)					
OP9Abs $C = torch.abs(A)$ OP10Power $C = torch.abs(A)$ OP11Exp $C = torch.abs(A)$ OP12Normalize $C = torch.abs(A)$ OP13ReLU $C = torch.abs(A)$ OP14Sign $C = torch.abs(A)$ OP15Heaviside $C = torch.abs(A)$ OP16Element-vise Invert $C = torch.sign(A)$ OP17Frobenius Norm $C = torch.norm(A, p='fro')$ OP18Determinant $C = torch.otex(A)$ OP19LogDeterminant $C = torch.abs(A)$ OP20SymEigRatio $A = A + A.T$ $e, v = torch.symeig(A, eigenvectors=False)$ $C = (c-1)/e[0]$ $A = A + A.T$ $e, v = torch.sum(A)/A.numel()$ OP21EigRatio $A = torch.sum(A)/A.numel()$ OP22Normalized Sum $C = torch.num(A)/A.numel()$ OP23L1 Norm $C = torch.num(A)/A.numel()$ OP24Hamming Distance $B = Heaviside(B)$ OP25KL Divergence $C = torch.num(A)/A.sumel(), -1)$ $A = A - reshape(A.shape[0], -1)$ $A = A - reshape(A.shape[0], -1)$ $A = A - reshape(A.shape[0], -1)$ $B = Reaviside(B)$ $OP24$ Hamming Distance $B = Heaviside(B)$ $OP26$ Cosine Similarity $C = torch.num(C)$ $OP27$ Softmax $C = torch.nes_like(A)$ $OP29$ Ones Like $C = torch.areos_like(A)$ $OP29$ Ones Like $C = torch.areos_like(A)$ $OP30$ Zeros Like $C = torch Careos_like(A)$ $OP31$ Less Than Zero $C = A$	OP8	AbsLog	$A[A==\cup] = 1$ C = torch.log(torch.abs(A))					
OP10Power $C = torch.pow(A, 2)$ OP11Exp $C = torch.exp(A)$ OP12Normalize $C = torch.exp(A)$ OP13ReLU $C = torch.functional.F.relu(A)$ OP14Sign $C = torch.functional.F.relu(A)$ OP15Heaviside $C = torch.ign(A)$ OP16Element-wise Invert $C = torch.norm(A, p='fro')$ OP17Frobenius Norm $C = torch.otext(A)$ OP18Determinant $C = torch.otext(A)$ OP19LogDeterminant $C = torch.igget(A)$ OP20SymEigRatio $A = A + A.T$ $e, v = torch.symeig(A, eigenvectors=False)$ $C = (e[-1]/e[0]$ $A = AA + A.T$ $e, v = torch.symeig(A, eigenvectors=False)$ $C = (e[-1]/e[0]$ $A = A reshape(A.shape[0], -1)$ $A = A + A.T$ $e, v = torch.sym(A)(A,numel(A)$ $e, v = torch.sym(A, A,numel(A, A))$ $e, v = torch.sym(A, A,numel(A, A))$ $e, v = torch.symeig(A, eigenvectors=False)$ $C = (e[-1]/e[0]$ $A = A + A.T$ $e, v = torch.symeig(A, eigenvectors=False)$ $C = (e[-1]/e[0]$ $A = A reshape(A.shape[0], -1)$ $A = A reshape(A.shape[0], -1)$ $A = torch.sym(A)(A, numel(A, A))$ $e, v = torch.sym(A, A, numel(A, A))$ $e, v = torch.sym(A, A)$ $OP21$ EigRatio $C = torch.n.KLDivLoss('batchmean')(A, B)$ $OP23$ Li Norm $C = torch.n.cosineSimilarity(A, B)$ $OP24$ Hamming Distance $B = Heaviside(B)$ $OP25$ KL Divergence $C = torch.n.cosineSimilarity(A, B)$ $OP26$ Cosine Similarity $C = torch.func$	OP9	Abs	C = torch.abs(A)					
OP11Exp $C = torch.exp(A)$ OP12Normalize $C = (a - A_{mecan})/A_{std}$ OP13ReLU $C = (a - A_{mecan})/A_{std}$ OP14Sign $C = torch.functional.F.relu(A)$ OP15Heaviside $C = torch.sign(A)$ OP16Element-wise Invert $C = torch.norm(A, p='fro')$ OP17Frobenius Norm $C = torch.loadet(A)$ OP19LogDeterminant $C = torch.loadet(A)$ OP19LogDeterminant $C [C!=C]=0$ $A = A + A T$ $e_v = torch.simeig(A, eigenvectors=False)$ $C = (c-1)/e[0]$ $A = torch.einsum('nc.mc-rum', [A,A])$ $e,v = torch.sum(A)/A.numel()$ $A = torch.sum(A)/A.numel()$ OP21EigRatio $C = torch.num(A)/A.numel()$ OP21EigRatio $C = torch.sum(A)/A.numel()$ OP22Normalized Sum $C = torch.sum(A)/A.numel()$ OP24Hamming Distance $B = Heaviside(B)$ OP25KL Divergence $C = torch.num.KLDivLoss('batchmean')(A,E)$ $A = A.reshape(A.shape[0], -1)$ $C = torch.sum(A)/A.numel()$ $A = Heaviside(B)$ $C = torch.num(C)$ OP25Sigmoid $C = torch.num(C)$ OP26Cosine Similarity $C = torch.sum(C)$ OP27Softmax $C = torch.sens_like(A)$ OP28Sigmoid $C = torch.sens_like(A)$ OP29Ones Like $C = torch.sens_like(A)$ OP30Zeros Like $C = torch cos_like(A)$ OP31Greater Than Zero $C = A > 0$ OP32Less Than Zero $C = A > 0$ OP33 <td>OP10</td> <td>Power</td> <td colspan="6">C = torch.pow(A, 2)</td>	OP10	Power	C = torch.pow(A, 2)					
OP12Normalize $C = (A - A_{mean})/A_{std}$ OP13ReLU $C[C!=C] = 0$ OP14Sign $C = torch.functional.F.relu(A)$ OP15Heaviside $C = torch.heaviside(A, values=[0])$ OP16Element-wise Invert $C = torch.norm(A, p='fro')$ OP17Frobenius Norm $C = torch.norm(A, p='fro')$ OP18Determinant $C = torch.logdet(A)$ OP19LogDeterminant $C = torch.logdet(A)$ OP20SymEigRatio $A = A.reshape(A.shape[0], -1)$ $A = AARA.T$ $A = AARA.T$ $A = AARA.T$ $A = A.reshape(A.shape[0], -1)$ $A = torch.einsum("nc.mc->nm", [A,A])$ $e, v = torch.eig(A)$ OP21EigRatio $C = (e[-1]/e[0])$ OP22Normalized Sum $C = (torch.nuch)/A.numel()$ OP23L1 Norm $torch.sum(abs(A))/A.numel()$ OP24Hamming Distance $C = torch.n.r.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(B.shape[0], -1)$ $B = Heaviside(B)$ OP25KL Divergence $C = torch.nuctional.F.sigmoid(A)$ OP26Cosine Similarity $C = torch.nuctional.F.softmax(A)$ OP27Softmax $C = torch.cosm_1ike(A)$ OP28Sigmoid $C = torch.cosm_1ike(A)$ OP29Ones Like $C = torch.cosm_1ike(A)$ OP31Greater Than Zero $C = A < 0$ OP32Less Than Zero $C = A < 0$ OP34Number Of Elements $C = torch.cosm_1ike(A)$	OP11	Exp	C = torch.exp(A)					
OP12Normalize $C[C!=C] = 0$ OP13ReLU $C = torch.functional.F.relu(A)$ OP14Sign $C = torch.functional.F.relu(A)$ OP15Heaviside $C = torch.sign(A)$ OP16Element-wise Invert $C = torch.neaviside(A, values=[0])$ OP17Frobenius Norm $C = torch.norm(A, p='fro')$ OP18Determinant $C = torch.logdet(A)$ OP19LogDeterminant $C = torch.logdet(A)$ OP20SymEigRatio $A = A + A.T$ $e, v = torch.einsum('nc,mc->nm', [A,A])$ $e, v = torch.einsum('nc,mc->nm', [A,A])$ $e, v = torch.einsum('nc,mc->nm', [A,A])$ $e, v = torch.sum(a)(A)/A.nume1()$ OP21EigRatio $C = torch.nuck(A)/A.nume1()$ OP22Normalized Sum $C = torch.sum(A)/A.nume1()$ OP23L1 Norm $C = torch.nuck(D)/A.nume1()$ OP24Hamming Distance $B = Heaviside(B)$ OP25KL Divergence $C = torch.nuck(D)/A.nume1()$ OP26Cosine Similarity $C = torch.nuc(C)$ OP27Softmax $C = torch.functional.F.softmax(A)$ OP28Sigmoid $C = torch.functional.F.softmax(A)$ OP29Ones Like $C = torch.cues_like(A)$ OP301Greater Than Zero $C = A > 0$ OP31Greater Than Zero $C = A > 0$ OP32Less Than Zero $C = torch Canser([A nume](D))$	0010		$C = (A - A_{magn})/A_{etd}$					
OP13 OP14 SignReLU Sign $C = torch.functional.F.relu(A)$ 	OP12	Normalize	C[C!=C] = 0					
OP14Sign $C = torch.sign(A)$ OP15Heaviside $C = torch.sign(A)$ OP16Element-wise Invert $C = 1/A$ OP17Frobenius Norm $C = torch.norm(A, p='fro')$ OP18Determinant $C = torch.det(A)$ OP19LogDeterminant $C = torch.logdet(A)$ OP20SymEigRatio $A = A.reshape(A.shape[0], -1)$ $A = AAA.T$ $A = AA.T$ $e, v = torch.symeig(A, eigenvectors=False)$ $C = e[-1]/e[0]$ $A = torch.einsum('nc,mc-nm', [A,A])$ $e, v = torch.sign(A)$ OP21EigRatio $C = torch.sum(A)A.numel()$ OP23L1 Norm $L1$ Norm $C = sum(A!=B)$ OP24Hamming DistanceOP25KL Divergence $C = torch.n.m.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ $A = A.reshape(A.shape[0], -1)$ $A = torch.sum(A)/A.numel()$ $A = torch.sum(A)/A.numel()$ $A = Heaviside(B)$ $C = torch.n.m.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ $B = B.reshape(B.shape[0], -1)$ $B = B.reshape(B.shape[0], -1)$ $B = B.reshape(B.shape[0], -1)$ $C = torch.sum(C)$ <t< td=""><td>OP13</td><td>ReLU</td><td>C = torch.functional.F.relu(A)</td></t<>	OP13	ReLU	C = torch.functional.F.relu(A)					
OP15 OP16Heaviside Element-wise Invert OP17 $C = torch.heaviside(A, values=[0])$ $C = 1/A$ OP17 OP18Determinant $C = torch.norm(A, p='fro')$ $C = torch.logdt(A)$ $C[C!=C]=0$ $A = A.reshape(A.shape[0], -1)$ $A = A@A.T$ OP20SymEigRatio $A = A + A.T$ $e, v = torch.symeig(A, eigenvectors=False)$ $C = [-1]/e[0]$ $A = A.reshape(A.shape[0], -1)$ $A = AA + A.T$ $e, v = torch.einsum('nc,mc->nm', [A,A])$ $e, v = torch.sym(A)/A.nume1()$ $C = torch.sum(A)/A.nume1()$ $C = torch.sum(A)/A.nume1()$ $C = torch.num(A)/A.nume1()$ OP21EigRatio $B = Heaviside(A)$ $C = (e[-1]/e[0])[0]$ $C = (e[-1]/e[0])[0]$ $C = torch.sum(A)/A.nume1()$ $A = Heaviside(A)$ OP24Hamming Distance $A = A - A.reshape(A.shape[0], -1)$ $A = Heaviside(B)$ $C = sum(A)=B)$ OP25KL Divergence $C = torch.nu.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ $B = B.reshape(B.shape[0], -1)$ $B = B.reshape(B.shape[0], -1)$ $C = torch.sum(C)$ OP27Softmax $C = torch.sum(C)$ $C = torch.sum(A)$ $C = torch.functional.F.sigmoid(A)$ $C = torch.sum(C)$ OP28Sigmoid $C = torch.czeros_like(A)$ $C = torch.sums[Like(A)]OP31Greater Than ZeroC = A < 0OP32Less Than ZeroC = torch Torcer(A nume1(D)]$	OP14	Sign	C = torch.sign(A)					
OP16 OP17 Probenius Norm $C = 1/A$ OP17 OP18Determinant $C = \text{torch.norm}(A, p='fro')$ OP19 OP19LogDeterminant $C = \text{torch.det}(A)$ OP19 OP20SymEigRatio $A = A \cdot \text{reshape}(A, \text{shape}[0], -1)$ $A = A0A.T$ $A = A + A.T$ $e, v = \text{torch.symeig}(A, eigenvectors=False)$ $C = (e[-1]/e[0]$ $A = A \cdot eshape(A, \text{shape}[0], -1)$ $A = \text{torch.einsum}('nc,mc->m'), [A,A])$ $e, v = \text{torch.eig}(A)$ $C = (e[-1]/e[0])$ $OP21$ EigRatio $A = \text{torch.eig}(A)$ $OP22$ Normalized Sum $C = (e[-1]/e[0])$ $OP23$ L1 Norm $C = \text{torch.sum}(A)/A.nume1()$ $OP24$ Hamming Distance $B = \text{Heaviside}(A)$ $OP24$ Hamming Distance $C = \text{sum}(A!=B)$ $OP25$ KL Divergence $C = \text{torch.nn.KLDivLoss('batchmean')}(A,B)$ $A = A.reshape(A.shape[0], -1)$ $B = B.reshape(A.shape[0], -1)$ $OP26$ Cosine Similarity $C = \text{torch.functional.F.softmax}(A)$ $OP27$ Softmax $C = \text{torch.functional.F.softmax}(A)$ $OP28$ Sigmoid $C = \text{torch.functional.F.sigmoid}(A)$ $OP30$ Zeros Like $C = \text{torch.surcs_like}(A)$ $OP31$ Greater Than Zero $C = A > 0$ $OP32$ Less Than Zero $C = \text{torch.Tencer}(A nume1()])$	OP15	Heaviside	C = torch.heaviside(A, values=[0])					
OP17 OP18Frobenius Norm Determinant $C = torch.norm(A, p='fro')$ $C = torch.logdet(A)$ $C[C!=C]=0$ $A = A.reshape(A.shape[0], -1)$ $A = A@A.T$ OP20SymEigRatio $A = A + A.T$ $e,v = torch.symeig(A, eigenvectors=False)$ $C = [-1]/e[0]$ $A = torch.eig(A)$ $C = torch.sum(inc,mc=nm', [A,A])$ $e,v = torch.sum(inc,mc=nm', [A,A])$ $e,v = torch.sum(A)/A.numel()$ OP21EigRatio $C = torch.sum(A)/A.numel()$ $C = torch.sum(A)/A.numel()$ OP22Normalized Sum L1 Norm $C = torch.nn.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ OP24Hamming Distance B = Heaviside(A)OP25KL Divergence $C = torch.nn.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ OP26Cosine Similarity $C = torch.nloticnal.F.sigmoid(A)$ $C = torch.sum(C)$ OP27Softmax Sigmoid OP30 $C = torch.sum(C)$ $C = torch.sum(C)$ OP30Zeros Like OP31 $C = torch.sume(A)$ $C = torch.sume(A)$ OP31Greater Than Zero OP32 $C = A<0$ $C = torch.sume(C)OP33Less Than ZeroC = A<0C = torch.rescr([A numel()])$	OP16	Element-wise Invert	C = 1/A					
OP18Determinant $C = torch.det(A)$ OP19LogDeterminant $C = torch.logdet(A)$ OP20SymEigRatio $A = A.reshape(A.shape[0], -1)$ $A = A + A.T$ $e, v = torch.symeig(A, eigenvectors=False)$ $C = e[-1]/e[0]$ $A = A + A.T$ $e, v = torch.eig(A)$ $e, v = torch.eig(A)$ OP21EigRatio $A = torch.einsum('nc,mc->nm', [A,A])$ $e, v = torch.eig(A)$ $C = (torch.sum(abs(A))/A.nume1()$ OP22Normalized Sum $C = torch.sum(A)/A.nume1()$ OP23L1 Norm $torch.sum(abs(A))/A.nume1()$ OP24Hamming Distance $B = Heaviside(B)$ OP25KL Divergence $C = torch.n.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ $B = B.reshape(B.shape[0], -1)$ OP26Cosine Similarity $C = torch.functional.F.sigmoid(A)$ OP27Softmax $C = torch.functional.F.sigmoid(A)$ OP28Sigmoid $C = torch.sum(C)$ OP30Zeros Like $C = torch.sum(ab(A))$ OP31Greater Than Zero $C = A>0$ OP32Less Than Zero $C = A<0$ OP34Number Of Elements $C = torch Tensor([A nume1()])$	OP17	Frobenius Norm	C = torch.norm(A, p='fro')					
OP19LogDeterminant $C = torch.logdet(A)$ $C[C!=C]=0$ $A = A.reshape(A.shape[0], -1)$ $A = A@A.T$ OP20SymEigRatio $A = A + A.T$ $e,v = torch.symeig(A, eigenvectors=False)C = e[-1]/e[0]A = A + A.Te,v = torch.einsum('nc,mc->nm', [A,A])e,v = torch.eig(A)C = (e[-1]/e[0])[0]OP21EigRatioC = (e[-1]/e[0])[0]C = torch.sum(a)/A.numel()C = torch.sum(abs(A))/A.numel()OP22Normalized SumL I NormC = torch.sum(A)/A.numel()A = Heaviside(A)OP24Hamming DistanceB = Heaviside(B)C = sum(A!=B)OP25KL DivergenceC = torch.nn.KLDivLoss('batchmean')(A,B)A = A.reshape(A.shape[0], -1)B = B.reshape(B.shape[0], -1)OP26Cosine SimilarityC = torch.nn.CosineSimilarity()(A, B)C = torch.sum(C)OP27SoftmaxSigmoidC = torch.sum(C)C = torch.sum(A)OP30Zeros LikeC = torch.nn.Zeros_like(A)OP31Greater Than ZeroOP32C = A < 0C = torch Zeros_([A nume1(1)])$	OP18	Determinant	C = torch.det(A)					
OP20SymEigRatio $A = A.reshape(A.shape[0], -1)$ $A = A@A.T$ OP20SymEigRatio $A = A + A.T$ $e,v = torch.symeig(A, eigenvectors=False)$ $C = e[-1]/e[0]$ $A = A.reshape(A.shape[0], -1)$ $A = torch.einsum('nc,mc->m', [A,A])$ $e,v = torch.eig(A)$ $C = (e[-1]/e[0]) [0]$ OP21OP21EigRatio $C = (e[-1]/e[0]) [0]$ $C = torch.eing(A) (A.numel())$ $C = torch.sum(A)/A.numel()$ $A = Heaviside(A)$ OP24Hamming Distance $B = Heaviside(A)$ $B = Heaviside(B)$ $C = sum(A!=B)$ OP25KL Divergence $C = torch.nn.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ $B = B.reshape(B.shape[0], -1)$ $C = torch.nn.CosineSimilarity()(A, B)$ $C = torch.sum(C)$ OP26Cosine Similarity $C = torch.sum(C)$ $C = torch.functional.F.softmax(A)$ $C = torch.ones_like(A)$ OP28Sigmoid $OP30$ $Zeros Like$ $C = torch.cros_like(A)$ $C = torch.ones_like(A)$ $C = torch.ones_like(A)$ OP31Greater Than Zero $OP33$ $Number Of ElementsC = torch Tansor([A numal()])$	OP19	LogDeterminant	C = torch.logdet(A)					
$A = A.reshape(A.shape(J, -1))$ $A = A@A.T$ $A = A@A.T$ $A = A@A.T$ $a = A + A.T$ $e,v = torch.symeig(A, eigenvectors=False)$ $C = e[-1]/e[O]$ $A = A.reshape(A.shape[O], -1)$ $A = torch.einsum('nc,mc->nm', [A,A])$ $e,v = torch.sum(A)/A.numel()$ $A = torch.sum(A)/A.numel()$ $OP23$ $L1$ Norm $C = torch.sum(A)/A.numel()$ $A = Heaviside(A)$ $B = Heaviside(B)$ $OP24$ $Hamming Distance$ $B = Heaviside(B)$ $C = sum(A!=B)$ $OP25$ $KL$ Divergence $C = sum(A!=B)$ $OP26$ $Cosine Similarity$ $DP26$ $Cosine Similarity$ $B = B.reshape(B.shape[O], -1)$ $B = B.reshape(B.shape[O], -1)$ $B = B.reshape(B.shape[O], -1)$ $C = torch.nn.CosineSimilarity()(A, B)$ $C = torch.nn.CosineSimilarity()(A, B)$ $C = torch.sum(C)$ $OP28$ $Sigmoid$ $OP29$ $Ones$ Like $C = torch.eros_like(A)$ $OP31$ $Greater Than Zero$ $OP32$ $Less Than Zero$ $C = AOP33Number OF Elements<$			C[C] = C] = 0					
OP20SymEigRatio $A = A + A.T$ OP21EigRatio $a = A + A.T$ OP21EigRatio $a = A + A.T$ OP21EigRatio $a = A \cdot reshape(A \cdot shape[0], -1)$ $A = torch \cdot einsum('nc, mc ->nm', [A,A])$ $e, v = torch \cdot einsum('nc, mc ->nm', [A,A])$ $OP21$ Normalized Sum $C = (e[-1]/e[0])[0]$ $OP23$ L1 Norm $c = torch \cdot sum(A)/A \cdot numel()$ $OP24$ Hamming Distance $B = Heaviside(A)$ $OP25$ KL Divergence $C = torch \cdot nn \cdot KLDivLoss('batchmean')(A,B)$ $A = A \cdot reshape(A \cdot shape[0], -1)$ $B = B \cdot reshape(B \cdot shape[0], -1)$ $OP26$ Cosine Similarity $C = torch \cdot nn \cdot Cosine Similarity()(A, B)$ $OP27$ Softmax $C = torch \cdot functional \cdot F \cdot sigmoid(A)$ $OP29$ Ones Like $C = torch \cdot ones_like(A)$ $OP30$ Zeros Like $C = torch \cdot zeros_like(A)$ $OP31$ Greater Than Zero $C = A < 0$ $OP32$ Less Than Zero $C = torch \cdot Tansor([A numel(D]))$			A = A.resnape(A.snape[0], -1)					
OP20Symplexities $A = A + A + A$ $e, v = torch.symeig(A, eigenvectors=False)$ $e, v = torch.symeig(A, eigenvectors=False)$ $C = e[-1]/e[0]$ $A = A.reshape(A.shape[0], -1)$ $A = torch.einsum('nc,mc->mm', [A,A])$ $e, v = torch.eig(A)$ $OP22$ Normalized Sum $OP23$ $L1$ Norm $OP24$ Hamming Distance $OP25$ $KL$ Divergence $OP26$ $Ooren Similarity$ $OP27$ $OP28$ $Op29$ $OP28$ $Sigmoid$ $OP29$ $OP30$ $Zeros$ Like $OP31$ $OP32$ $Less$ Than Zero $OP33$ $Number Of Elements$	0220	SumFigDatio	$A = A \oplus A$ . I					
OP21EigRatio $C = e[-1]/e[0]$ $A = A.reshape(A.shape[0], -1)$ $A = torch.einsum('nc,mc->nm', [A,A])$ $e, v = torch.sum(A)/A.numel()$ $OP23$ $L1$ Norm $OP24$ Hamming Distance $OP25$ $KL$ Divergence $C = sum(A!=B)$ $OP26$ $Cosine Similarity$ $OP26$ $Cosine Similarity$ $OP26$ $Ocsine Similarity$ $OP27$ $Softmax$ $OP28$ $Sigmoid$ $OP29$ $Ones Like$ $OP30$ $Zeros Like$ $OP31$ $OP32$ $Less$ Than Zero $OP33$ $Number Of Elements$ $OP33$ $Number Of Elements$	0120	SymElgRatio	$A = A + A \cdot I$					
OP21EigRatio $A = A.reshape[0], -1)$ $A = torch.einsum('nc,mc->nm', [A,A])$ $e,v = torch.eig(A)$ $C = (e[-1]/e[0])[0]$ $C = torch.sum(A)/A.numel()$ $torch.sum(abs(A))/A.numel()$ $A = Heaviside(A)$ $B = Heaviside(B)$ $C = sum(A!=B)$ $C = torch.nn.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ $B = B.reshape(B.shape[0], -1)$ $C = torch.nn.CosineSimilarity()(A, B)$ $C = torch.sum(C)$ $C = torch.sum(C)$ OP27Softmax OP28 Sigmoid OP29C = torch.functional.F.sigmoid(A) $C = torch.sum(C)$ $C = torch.sum(C)$ OP29Ones Like OP30 Qeros LikeC = torch.cres_like(A) $C = torch.zeros_like(A)$ $C = torch.zeros_like(A)OP30Zeros LikeOP31C = torch.cres_like(A)C = torch.Tan ZeroC = torch.Tan ZeroC = A<0$			$C = \rho[-1]/\rho[0]$					
OP21EigRatio $A = torch.einsum('nc,mc->nm', [A,A])$ $e,v = torch.eig(A)$ $C = (e[-1]/e[0])[0]$ $C = torch.sum(A)/A.numel()$ $A = Heaviside(A)$ OP24Hamming Distance $B = Heaviside(A)$ $B = Heaviside(B)$ $C = sum(A!=B)$ OP25KL Divergence $C = torch.nn.KLDivLoss('batchmean')(A,B)$ $A = A.reshape(A.shape[0], -1)$ $B = B.reshape(B.shape[0], -1)$ OP26Cosine Similarity $C = torch.nn.CosineSimilarity()(A, B)$ $C = torch.sum(C)$ OP27Softmax Sigmoid $C = torch.functional.F.softmax(A)$ $C = torch.ones_like(A)$ OP28Sigmoid Greater Than Zero OP31 $C = A < O$ $C = torch Tensor([A nume]()])$			$A = A \cdot reshape(A \cdot shape[0], -1)$					
OP21EigRatioOP22Normalized SumOP23L1 NormOP24Hamming DistanceOP25KL DivergenceOP26Cosine SimilarityOP27SoftmaxOP28SigmoidOP29Ones LikeOP20Cersot LikeOP21Greater Than ZeroOP32Less Than ZeroOP33Number Of Elements			A = torch.einsum('nc.mc->nm', [A.A])					
OP22Normalized Sum $C = (e[-1]/e[0])[0]$ OP23L1 Norm $C = torch.sum(A)/A.numel()$ OP24Hamming Distance $torch.sum(abs(A))/A.numel()$ OP25KL Divergence $C = sum(A!=B)$ OP26Cosine Similarity $C = torch.nn.KLDivLoss('batchmean')(A,B)$ OP27Softmax $C = torch.nn.CosineSimilarity()(A, B)$ OP28Sigmoid $C = torch.functional.F.softmax(A)$ OP29Ones Like $C = torch.cnes_like(A)$ OP31Greater Than Zero $C = A < 0$ OP33Number Of Elements $C = torch.Tansor([A nume1()])$	OP21	EigRatio	e,v = torch.eig(A)					
OP22Normalized SumC = torch.sum(A)/A.numel()OP23L1 Normtorch.sum(abs(A))/A.numel()OP24Hamming DistanceB = Heaviside(A)OP25KL DivergenceC = torch.nn.KLDivLoss('batchmean')(A,B)OP26Cosine SimilarityC = torch.nn.KLDivLoss('batchmean')(A,B)OP27SoftmaxC = torch.nn.CosineSimilarity()(A, B)OP28SigmoidC = torch.functional.F.softmax(A)OP29Ones LikeC = torch.ones_like(A)OP30Zeros LikeC = torch.zeros_like(A)OP31Greater Than ZeroC = A <o< td="">OP33Number Of ElementsC = torch Tansor([A nume]()])</o<>			C = (e[-1]/e[0])[0]					
OP23L1 Normtorch.sum(abs(A))/A.numel()OP24Hamming Distancea = Heaviside(A)OP25KL DivergenceC = sum(A!=B)OP26Cosine SimilarityC = torch.nn.KLDivLoss('batchmean')(A,B)A = A.reshape(A.shape[0], -1)B = B.reshape(B.shape[0], -1)OP26Cosine SimilarityC = torch.nn.CosineSimilarity()(A, B)OP27SoftmaxC = torch.functional.F.softmax(A)OP28SigmoidC = torch.functional.F.sigmoid(A)OP29Ones LikeC = torch.ones_like(A)OP30Zeros LikeC = torch.zeros_like(A)OP31Greater Than ZeroC = A<0	OP22	Normalized Sum	C = torch.sum(A)/A.numel()					
OP24Hamming DistanceA = Heaviside(A)OP25KL DivergenceB = Heaviside(B)OP26Cosine SimilarityC = torch.nn.KLDivLoss('batchmean')(A,B)A = A.reshape(A.shape[0], -1)B = B.reshape(B.shape[0], -1)OP26Cosine SimilarityC = torch.nn.CosineSimilarity()(A, B)OP27SoftmaxC = torch.functional.F.softmax(A)OP28SigmoidC = torch.functional.F.sigmoid(A)OP29Ones LikeC = torch.ones_like(A)OP30Zeros LikeC = torch.zeros_like(A)OP31Greater Than ZeroC = A<0	OP23	L1 Norm	<pre>torch.sum(abs(A))/A.numel()</pre>					
OP24Hamming DistanceB = Heaviside(B) C = sum(A!=B)OP25KL DivergenceC = torch.nn.KLDivLoss('batchmean')(A,B) A = A.reshape(A.shape[0], -1) B = B.reshape(B.shape[0], -1) C = torch.nn.CosineSimilarity()(A, B) C = torch.sum(C)OP27Softmax OP28C = torch.functional.F.softmax(A) C = torch.functional.F.sigmoid(A) C = torch.ones_like(A) C = torch.zeros_like(A)OP30Zeros Like OP31C = torch.cones_like(A) C = A<0			A = Heaviside(A)					
OP25KL DivergenceC = sum(A!=B) C = torch.nn.KLDivLoss('batchmean')(A,B) A = A.reshape(A.shape[0], -1) B = B.reshape(B.shape[0], -1) C = torch.nn.CosineSimilarity()(A, B) C = torch.sum(C)OP27Softmax OP28C = torch.functional.F.softmax(A) C = torch.functional.F.sigmoid(A) C = torch.ones_like(A)OP29Ones Like OP30 OP31C = torch.ones_like(A) C = torch.zeros_like(A) C = torch.zeros_like(A)OP33Number Of ElementsC = torch Tensor([A nume]()])	OP24	Hamming Distance	B = Heaviside(B)					
OP25KL DivergenceC = torch.nn.KLDivLoss('batchmean')(A,B)OP26Cosine SimilarityA = A.reshape(A.shape[0], -1)OP26Cosine SimilarityC = torch.nn.CosineSimilarity()(A, B)OP27SoftmaxC = torch.nn.CosineSimilarity()(A, B)OP28SigmoidC = torch.functional.F.softmax(A)OP29Ones LikeC = torch.functional.F.sigmoid(A)OP30Zeros LikeC = torch.creos_like(A)OP31Greater Than ZeroC = A>OOP33Number Of ElementsC = torch.Tensor([A nume]()])			C = sum(A!=B)					
OP26Cosine SimilarityA = A.reshape(A.shape[0], -1) B = B.reshape(B.shape[0], -1) C = torch.nn.CosineSimilarity()(A, B) C = torch.sum(C)OP27SoftmaxC = torch.nn.CosineSimilarity()(A, B) C = torch.sum(C)OP28SigmoidC = torch.functional.F.softmax(A) C = torch.functional.F.sigmoid(A)OP29Ones LikeC = torch.ones_like(A) C = torch.zeros_like(A)OP30Zeros LikeC = torch.ceros_like(A) C = torch.zeros_like(A)OP31Greater Than Zero OP32C = A<0 C = A<0	OP25	KL Divergence	C = torch.nn.KLDivLoss('batchmean')(A,B)					
OP26Cosine SimilarityB = B.reshape(B.shape[0], -1)OP27Cosine SimilarityC = torch.nn.CosineSimilarity()(A, B)OP28SigmoidC = torch.sum(C)OP29Ones LikeC = torch.functional.F.sigmoid(A)OP30Zeros LikeC = torch.ones_like(A)OP31Greater Than ZeroC = A<0			A = A.reshape(A.shape[0], -1)					
OP27SoftmaxC = torch.nn.CosineSimilarity()(A, B)OP28SigmoidC = torch.sum(C)OP29Ones LikeC = torch.functional.F.sigmoid(A)OP30Zeros LikeC = torch.ones_like(A)OP31Greater Than ZeroC = A<0	OP26	Cosine Similarity	B = B.reshape(B.shape[0], -1)					
OP27SoftmaxC = torch.sum(C)OP28SigmoidC = torch.functional.F.softmax(A)OP29Ones LikeC = torch.functional.F.sigmoid(A)OP30Zeros LikeC = torch.ones_like(A)OP31Greater Than ZeroC = A>0OP32Less Than ZeroC = A<0		5	C = torch.nn.CosineSimilarity()(A, B)					
OF27SoftmaxC = torch.functional.F.softmax(A)OP28SigmoidC = torch.functional.F.sigmoid(A)OP29Ones LikeC = torch.ones_like(A)OP30Zeros LikeC = torch.ones_like(A)OP31Greater Than ZeroC = A>0OP32Less Than ZeroC = A<0	0027	Softmax	C = torch.sum(C)					
OP29Ones LikeC = torch.runctional.r.signoid(A)OP30Zeros LikeC = torch.ones_like(A)OP31Greater Than ZeroC = A>0OP32Less Than ZeroC = A<0	OP29	Sigmoid	C = torcn.iunctional.F.soItmax(A)					
OP30Zeros Like $C = torch.snes_like(A)$ OP31Greater Than Zero $C = A>0$ OP32Less Than Zero $C = A<0$ OP33Number Of Elements $C = torch.zeros_like(A)$	OP20	Ones Like	<pre>C = torch.lunctional.F.Sigmold(A) C = torch energ like(A)</pre>					
OP31Greater Than Zero $C = A>0$ OP32Less Than Zero $C = A<0$ OP33Number Of Elements $C = torch Tensor([A nume]()])$	OP20	Zeros Like	$C = torch.ones_tike(A)$					
$\begin{array}{c} OF ST \\ OP 32 \\ OP 33 \\$	OP31	Greater Than Zero	$C = \Delta > 0$					
$\begin{array}{c c} OF 32 \\ OP 33 \\ Number Of Flements \\ OP 33 \\ $	OP32	Less Than Zero	C = A < 0					
$\mathbf{x}_{i}$	OP33	Number Of Flements	C = torch Tensor([A nume]()])					

Table 1: List of operations available for the genetic program.