Supplementary Materials: Adaptive Neural Trees

A. Training algorithm

Algorithm 1 ANT Optimisation

```
Initialise topology $\mathbb{T}$ and parameters $\mathbb{O}$    ▷ $\mathbb{T}$ is set to a root node with one solver and one transformer
Optimise parameters in $\mathbb{O}$ via gradient descent on NLL    ▷ Learning root classifier
Set the root node “suboptimal”
while true do    ▷ Growth of $\mathbb{T}$ begins
    Freeze all parameters $\mathbb{O}$
    Pick next “suboptimal” leaf node $l \in \mathcal{N}_{leaf}$ in the breadth-first order
    Add (1) router to $l$ and train new parameters    ▷ Split data
    Add (2) transformer to $l$ and train new parameters    ▷ Deepen transform
    Add (1) or (2) to $\mathbb{T}$ if validation error decreases, otherwise set $l$ to “optimal”
    Add any new modules to $\mathbb{O}$
    if no “suboptimal” leaves remain then
        Break
Unfreeze and train all parameters in $\mathbb{O}$    ▷ Global refinement with fixed $\mathbb{T}$
```

B. Additional related work

Here we provide an expanded review of related works, precluded from the main text due to space limit. The tree-structure of ANTs naturally performs conditional computation. We can also view the proposed tree-building algorithm as a form of neural architecture search. We provide surveys of these areas and their relations to ANTs.

Conditional computation: in NNs, computation of each sample engages every parameter of the model. In contrast, DTs route each sample to a single path, only activating a small fraction of the model. Bengio (2013) advocated for this notion of conditional computation to be integrated into NNs, and this has become a topic of growing interest. Rationales for using conditional computation ranges from attaining better capacity-to-computation ratio (Bengio et al., 2013; Davis & Arel, 2013; Bengio et al., 2015; Shazeer et al., 2017) to adapting the required computation to the difficulty of the input and task (Bengio et al., 2015; Almahairi et al., 2016; Teerapittayanon et al., 2016; Graves, 2016; Figurnov et al., 2017; Veit & Belongie, 2017). We view the growth procedure of ANTs as having a similar motivation with the latter—processing raw pixels is suboptimal for computer vision tasks, but we have no reason to believe that the hundreds of convolutional layers in current state-of-the-art architectures (He et al., 2016; Huang et al., 2017) are necessary either. Growing ANTs adapts the architecture complexity to the dataset as a whole, with routers determining the computation needed on a per-sample basis.

Neural architecture search: the ANT growing procedure is related to the progressive growing of NNs (Fahlman & Lebiere, 1990; Hinton et al., 2006; Xiao et al., 2014; Chen et al., 2016; Srivastava et al., 2015; Lee et al., 2017; Cai et al., 2018; İrsoy & Alpaydın, 2018), or more broadly, the field of neural architecture search (Zoph & Le, 2017; Brock et al., 2017; Cortes et al., 2017). This approach, mainly via greedy layerwise training, has historically been one solution to optimising NNs (Fahlman & Lebiere, 1990; Hinton et al., 2006). However, nowadays it is possible to train NNs in an end-to-end fashion. One area which still uses progressive growing is lifelong learning, in which a model needs to adapt to new tasks while retaining performance on previous ones (Xiao et al., 2014; Lee et al., 2017). In particular, Xiao et al. (2014) introduced a method that grows a tree-shaped network to accommodate new classes. However, their method never transforms the data before passing it to the children classifiers, and hence never benefit from the parent's representations.

Whilst we learn the architecture of an ANT in a greedy, layerwise fashion, several other methods search globally. Based on a variety of techniques, including evolutionary algorithms (Stanley & Miikkulainen, 2002; Real et al., 2017), reinforcement learning (Zoph & Le, 2017), sequential optimisation (Liu et al., 2017) and boosting (Cortes et al., 2017), these methods find extremely high-performance yet complex architectures. In our case, we constrain the search space to simple tree-structured NNs, retaining desirable properties of DTs such as data-dependent computation and interpretable structures, while keeping the space and time requirement of architecture search tractable thanks to the locality of our growth procedure.

Cascaded trees and forests: another noteworthy strand of work for feature learning with tree-structured models is cascaded forests—stacks of RFs where the outputs of intermediate models are fed into the subsequent ones (Montillo et al., 2011; Kontschieder et al., 2013; Zhou & Feng, 2017). It has been shown how a cascade of DTs can be mapped to NNs with sparse connections (Sethi, 1990), and more recently Richmond et al. (2015) extended this argument to RFs. However, the features obtained in this approach are the intermediate outputs of respective component models, which are not optimised for the target task, and cannot be learned end-to-end, thus limiting its representational quality. Recently, Feng et al. (2018) introduced a method to jointly train a cascade of gradient boosted trees (GBTs) to improve the limited representation learning ability of such previous work. A variant of target propagation (Lee et al., 2015) was designed to enable the end-to-end training of cascaded GBTs, each of which is non-differentiable and thus not amenable to back-propagation.

C. Set-up details

Data: we perform our experiments on the SARCOS robot inverse dynamics dataset[1], the MNIST digit classification task (LeCun et al., 1998) and the CIFAR-10 object recognition task (Krizhevsky & Hinton, 2009). The SARCOS dataset consists of 44,484 training and 4,449 testing examples, where the goal is to map from the 21-dimensional input space (7 joint positions, 7 joint velocities and 7 joint accelerations) to the corresponding 7 joint torques (Vijayakumar & Schaal, 2000). No dataset preprocessing or augmentation is used. The MNIST dataset consists of $60,000$ training and $10,000$ testing examples, all of which are 28×28 grayscale images of digits from 0 to 9 (10 classes). The dataset is preprocessed by subtracting the mean, but no data augmentation is used. The CIFAR-10 dataset consists of $50,000$ training and $10,000$ testing examples, all of which are 32×32 coloured natural images drawn from 10 classes. We adopt an augmentation scheme widely used in the literature (Goodfellow et al., 2013; Lin et al., 2014; Springenberg et al., 2015; He et al., 2016; Huang et al., 2017) where images are zero-padded with 4 pixels on each side, randomly cropped and horizontally mirrored. For all three datasets, we hold out 10% of training images as a validation set. The best model is selected based on the validation accuracy over the course of ANT training, spanning both the growth phase and the refinement phase, and its test accuracy is reported.

Training: both the growth phase and the refinement phase of ANTs are performed on a single Titan X GPU on all three datasets. For all the experiments in this paper, we employ the following training protocol: (1) optimise parameters using Adam (Kingma & Ba, 2014) with initial learning rate of 10^{-3} and $\beta = [0.9, 0.999]$, with minibatches of size 512; (2) during the growth phase, employ early stopping with a patience of 5, that is, training is stopped after 5 epochs of no progress on the validation set; (3) during the refinement phase, train for 300 epochs for SARCOS, 100 epochs for MNIST and 200 epochs for CIFAR-10, decreasing the learning rate by a factor of 10 at every multiple of 50. Training times are provided in Supp. Sec. D.

We observe that the patience level is an important hyperparameter which affects the quality of the growth phase; very low or high patience levels result in new modules underfitting or overfitting locally, thus preventing meaningful further growth and limiting the accuracy of the resultant models. We tuned this hyperparameter using the validation sets, and set the patience level to 5, which produced consistently good performance on SARCOS, MNIST and CIFAR-10 datasets across different specifications of primitive modules. A quantitative evaluation on CIFAR-10 is given in Supp. Sec. E.

[1]http://www.gaussianprocess.org/gpml/data/

In the SARCOS experiments, all the non-NN-based methods were trained using scikit-learn (Pedregosa et al., 2011). Hidden layers in the baseline MLPs are followed by tanh non-linearities and contain 256 units to be consistent with the complexity of transformer modules.

Primitive modules: we train ANTs with a range of primitive modules as shown in Tab. 2 in the main text. For simplicity, we define the modules based on three types of NN layers: convolutional, global-average-pooling (GAP) and fully-connected (FC). Solver modules are fixed as linear models e.g. linear classifier and linear regression. Router modules are binary classifiers with a sigmoid output. All convolutional and FC layer are followed by ReLU or tanh non-linearities, except in the last layers of solvers and routers. For image classification experiments, we also apply 2×2 max-pooling to feature maps after every d transformer modules where d is the downsample frequency. For the SARCOS regression experiment, hidden layers in the routers and transformers contain 256 units. We balance the number of parameters in the router and transformer modules to be of the same order of magnitude to avoid favouring either partitioning the data or learning more expressive features.

D. Training times

Tab. 1 summarises the time taken on a single Titan X GPU for the growth phase and refinement phase of various ANTs, and compares against the training time of All-CNN (Springenberg et al., 2015). Local optimisation during the growth phase means that the gradient computation is constrained to the newly added component of the graph, allowing us to grow a good candidate model under 3 hours on one GPU.

Table 1: Training time comparison. Time and number of epochs taken for the growth and refinement phase are shown. along with the time required to train the baseline, All-CNN (Springenberg et al., 2015).

	Growth		**Fine-tune**	
Model	Time	Epochs	Time	Epochs
All-CNN (baseline)	–	–	1.1 (hr)	200
ANT-CIFAR10-A	1.3 (hr)	236	1.5 (hr)	200
ANT-CIFAR10-B	0.8 (hr)	313	0.9 (hr)	200
ANT-CIFAR10-C	0.7 (hr)	285	0.8 (hr)	200

E. Effect of training steps in the growth phase

Fig. 1 compares the validation accuracies of the same ANT-CIFAR-C model trained on the CIFAR-10 dataset with varying levels of patience during early stopping in the growth phase. A higher patience level corresponds to more training epochs for optimising new modules in the growth phase. When the patience level is 1, the architecture growth terminates prematurely and plateaus at low accuracy at 80%. On the other hand, a patience level of 15 causes the model to overfit locally with 87%. The patience level of 5 gives the best results with 91% validation accuracy.

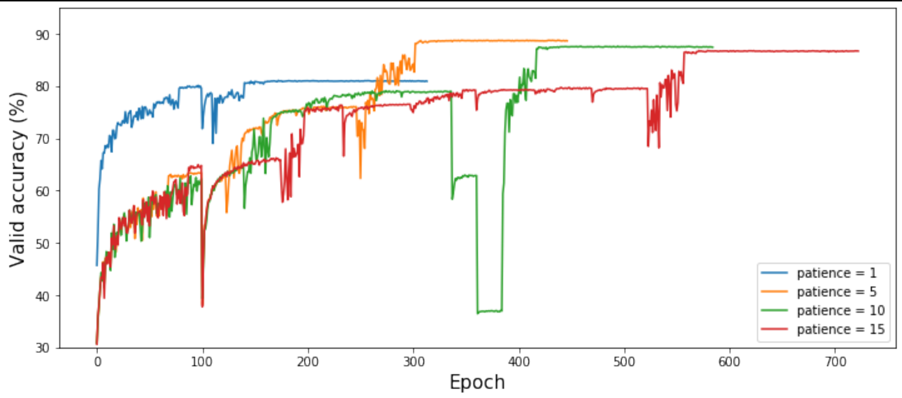

Figure 1: Effect of patience level on the validation accuracy trajectory during training. Each curve shows the validation accuracy on CIFAR-10 dataset.

F. Expert specialisation

We investigate if the learned routing strategy is meaningful by comparing the classification accuracy of our default path-wise inference against that of the predictions from the leaf node with the smallest reaching probability. Tab. 2 shows that using the least likely "expert" leads to a substantial drop in classification accuracy, down to close to that of random guess or even worse for large trees (ANT-MNIST-C and ANT-CIFAR10-C). This demonstrates that features in ANTs become specialised to the subsets of the partitioned input space at lower levels in the tree hierarchy.

Table 2: Comparison of classification performance between the default single-path inference scheme and the prediction based on the least likely expert. between the

Module Spec.	**Error %**	**Error %**
	(Selected path)	(Least likely path)
ANT-MNIST-A	0.69	86.18
ANT-MNIST-B	0.73	81.98
ANT-MNIST-C	1.68	98.84
ANT-CIFAR10-A	8.32	74.28
ANT-CIFAR10-B	9.18	89.74
ANT-CIFAR10-C	9.34	97.52

G. Visualisation of discovered architectures

Fig. 2 shows ANT architectures discovered on the MNIST (i-iii) and CIFAR-10 (iv-vi) datasets. We observe three notable trends. Firstly, a large proportion of the learned routers separate examples based on their classes (red histograms) with very high confidence (blue histograms). The ablation study in Sec. 5. 1 (Tab. 4 in the main text) shows that such hierarchical clustering benefits predictive performance, while the conditional computation enables more lightweight inference (Tab. 3 in the main text). Secondly, most architectures learn a few levels of features before resorting to primarily splits. However, over half of the architectures (ii-v) still learn further representations beyond the first split. Secondly, all architectures are unbalanced. This reflects the fact that some groups of samples may be easier to classify than others. This property is reflected by traditional DT algorithms, but not "neural" tree-structured models with pre-specified architectures (Laptev & Buhmann, 2014; Frosst & Hinton, 2017; Kontschieder et al., 2015; Ioannou et al., 2016).

H. FLOPS

Tab.3 reports the floating point operations per second (FLOPS) of ANT models for two inference schemes. The results for ResNet110 and DenseNet were retrieved from (Guan et al., 2017) and (Huang et al., 2018), respectively. The FLOPs of all other models were computed using TorchStat toolbox available at `https://github.com/Swall0w/torchstat`. Using single-path inference reduces FLOPS in all ANT models to varying degrees.

Table 3: Comparison of FLOPs.

	Model	**FLOPS**	**FLOPS**
		(multi-path)	(single-path)
MNIST	Linear Classifier	8K	-
	LeNet-5	231 K	-
	ANT-MNIST-C	99K	83K
	ANT-MNIST-B	346K	331K
	ANT-MNIST-A	382K	380K
CIFAR-10	Net-in-Net	222M	-
	All-CNN	245M	-
	ResNet-110	256M	-
	DenseNet-BC (k=24)	9388M	-
	ANT-CIFAR10-C	66M	61M
	ANT-CIFAR10-B	163M	149M
	ANT-CIFAR10-A	254M	243M

I. Ensembling

As with traditional DTs (Breiman, 2001) and NNs (Hansen & Salamon, 1990), ANTs can be ensembled to gain improved performance. In Tab. 4 we show the results of ensembling 8 ANTs (using the "-A" configurations for classification), each of which is trained with a randomly chosen split between training and validation sets. We compare against the single tree models, trained with the default split. In all cases both the multi-path and single-path inference performance is noticeably improved, and in MNIST we reach close to state-of-the-art performance (0.29% versus 0.25% (Sabour et al., 2017)) with significantly fewer parameters (851k versus 8.2M).

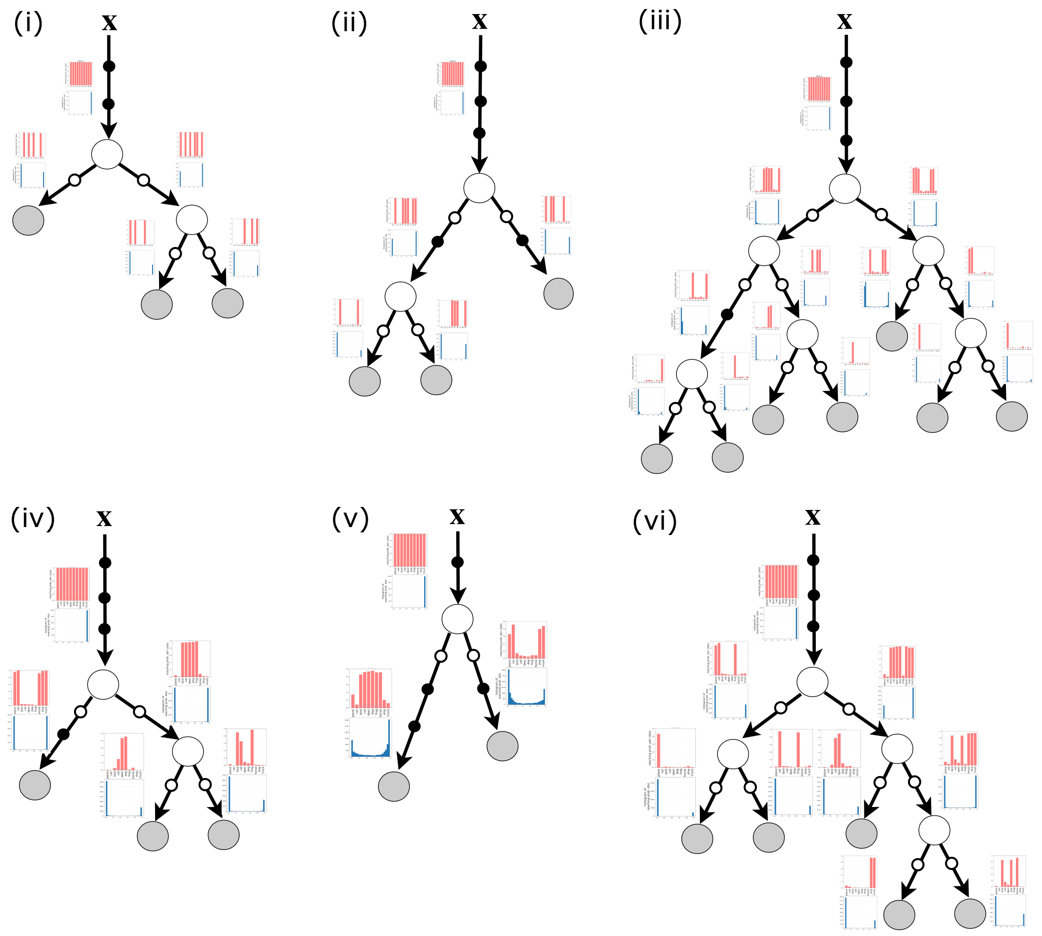

Figure 2: Illustration of discovered ANT architectures. (i) ANT-MNIST-A, (ii) ANT-MNIST-B, (iii) ANT-MNIST-C, (iv) ANT-CIFAR10-A, (v) ANT-CIFAR10-B, (vi) ANT-CIFAR10-C. Histograms in red and blue show the class distributions and path probabilities at respective nodes. Small black circles on the edges represent transformers, circles in white at the internal nodes represent routers, and circles in gray are solvers. The small white circles on the edges denote specific cases where transformers are identity functions.

Table 4: Comparison of prediction errors of a single ANT versus an ensemble of 8, with predictions averaged over all ANTs in the ensemble.

	MNIST (Class Error %)		**CIFAR-10** (Class Error %)		**SARCOS** (MSE)	
	Multi-path	Single-path	Multi-path	Single-path	Multi-path	Single-path
Single model	0.64	0.69	8.31	8.32	1.384	1.542
Ensemble	0.29	0.30	7.76	7.79	1.226	1.372

Table 5: Parameter counts for a single ANT versus an ensemble of 8.

	MNIST (No. Params.)		**CIFAR-10** (No. Params.)		**SARCOS** (No. Params.)	
	Multi-path	Single-path	Multi-path	Single-path	Multi-path	Single-path
Single model	100,596	84,935	1.4M	1.0M	103,823	61,640
Ensemble	850,775	655,449	8.7M	7.4M	598,280	360,766

References

Almahairi, A., Ballas, N., Cooijmans, T., Zheng, Y., Larochelle, H., and Courville, A. Dynamic capacity networks. In *ICML*, 2016.

Bengio, E., Bacon, P.-L., Pineau, J., and Precup, D. Conditional computation in neural networks for faster models. *CoRR*, 2015.

Bengio, Y. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pp. 1–37. Springer, 2013.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, 2013.

Breiman, L. Random forests. *Machine learning*, 45(1): 5–32, 2001.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. Smash: one-shot model architecture search through hypernetworks. *CoRR*, 2017.

Cai, H., Chen, T., Zhang, W., Yu, Y., and Wang, J. Efficient architecture search by network transformation. In *AAAI*, 2018.

Chen, T., Goodfellow, I., and Shlens, J. Net2net: Accelerating learning via knowledge transfer. In *ICLR*, 2016.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., and Yang, S. Adanet: Adaptive structural learning of artificial neural networks. In *ICML*, pp. 874–883, 2017.

Davis, A. and Arel, I. Low-rank approximations for conditional feedforward computation in deep neural networks. *CoRR*, 2013.

Fahlman, S. E. and Lebiere, C. The cascade-correlation learning architecture. In *Advances in neural information processing systems*, pp. 524–532, 1990.

Feng, J., Yu, Y., and Zhou, Z.-H. Multi-layered gradient boosting decision trees. In *Advances in Neural Information Processing Systems*, pp. 3551–3561, 2018.

Figurnov, M., Collins, M. D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D. P., and Salakhutdinov, R. Spatially adaptive computation time for residual networks. *CVPR*, pp. 1790–1799, 2017.

Frosst, N. and Hinton, G. E. Distilling a neural network into a soft decision tree. *CoRR*, 2017.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout networks. In *ICML*, 2013.

Graves, A. Adaptive computation time for recurrent neural networks. *CoRR*, abs/1603.08983, 2016.

Guan, J., Liu, Y., Liu, Q., and Peng, J. Energy-efficient amortized inference with cascaded deep classifiers. *arXiv preprint arXiv:1710.03368*, 2017.

Hansen, L. K. and Salamon, P. Neural network ensembles. *IEEE TPAMI*, 12(10):993–1001, 1990.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, 18 (7):1527–1554, 2006.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. Densely connected convolutional networks. In *CVPR*, 2017.

Huang, G., Liu, S., Van der Maaten, L., and Weinberger, K. Q. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2752–2761, 2018.

Ioannou, Y., Robertson, D., Zikic, D., Kontschieder, P., Shotton, J., Brown, M., and Criminisi, A. Decision forests, convolutional networks and the models in-between. *CoRR*, 2016.

İrsoy, O. and Alpaydın, E. Continuously constructive deep neural networks. *arXiv preprint arXiv:1804.02491*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, 2014.

Kontschieder, P., Kohli, P., Shotton, J., and Criminisi, A. Geof: Geodesic forests for learning coupled predictors. In *CVPR*, 2013.

Kontschieder, P., Fiterau, M., Criminisi, A., and Rota Bulo, S. Deep neural decision forests. In *ICCV*, pp. 1467–1475, 2015.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Laptev, D. and Buhmann, J. M. Convolutional decision trees for feature learning and segmentation. In *German Conference on Pattern Recognition*, pp. 95–106. Springer, 2014.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pp. 498–515. Springer, 2015.

Lee, J., Yun, J., Hwang, S., and Yang, E. Lifelong learning with dynamically expandable networks. *CoRR*, 2017.

Lin, M., Chen, Q., and Yan, S. Network in network. In *ICLR*, 2014.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. *CoRR*, 2017.

Montillo, A., Shotton, J., Winn, J., Iglesias, J. E., Metaxas, D., and Criminisi, A. Entangled decision forests and their application for semantic segmentation of ct images. In *Biennial International Conference on Information Processing in Medical Imaging*, pp. 184–196. Springer, 2011.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *JMLR*, 12(Oct):2825–2830, 2011.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Le, Q., and Kurakin, A. Large-scale evolution of image classifiers. *CoRR*, 2017.

Richmond, D. L., Kainmueller, D., Yang, M., Myers, E. W., and Rother, C. Mapping stacked decision forests to deep and sparse convolutional neural networks for semantic segmentation. *CoRR*, 2015.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pp. 3856–3866, 2017.

Sethi, I. K. Entropy nets: From decision trees to neural networks. *Proceedings of the IEEE*, pp. 1605–1613, 1990.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ICLR*, abs/1701.06538, 2017.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *CoRR*, 2015.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Stanley, K. O. and Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 2002.

Teerapittayanon, S., McDanel, B., and Kung, H. T. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, 2016.

Veit, A. and Belongie, S. Convolutional networks with adaptive computation graphs. *CoRR*, 2017.

Vijayakumar, S. and Schaal, S. Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space. In *ICML*, volume 1, pp. 288–293, 2000.

Xiao, T., Zhang, J., Yang, K., Peng, Y., and Zhang, Z. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *ACM Multimedia*, 2014.

Zhou, Z.-H. and Feng, J. Deep forest: Towards an alternative to deep neural networks. In *IJCAI*, 2017.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *ICLR*, 2017.