tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

"The lottery ticket hypothesis for gigantic pre-trained models"

Atlas Wang – UT Austin

March 30, 2021

www.tinyML.org

tinyML Talks Sponsors

tinyML Strategic Partner

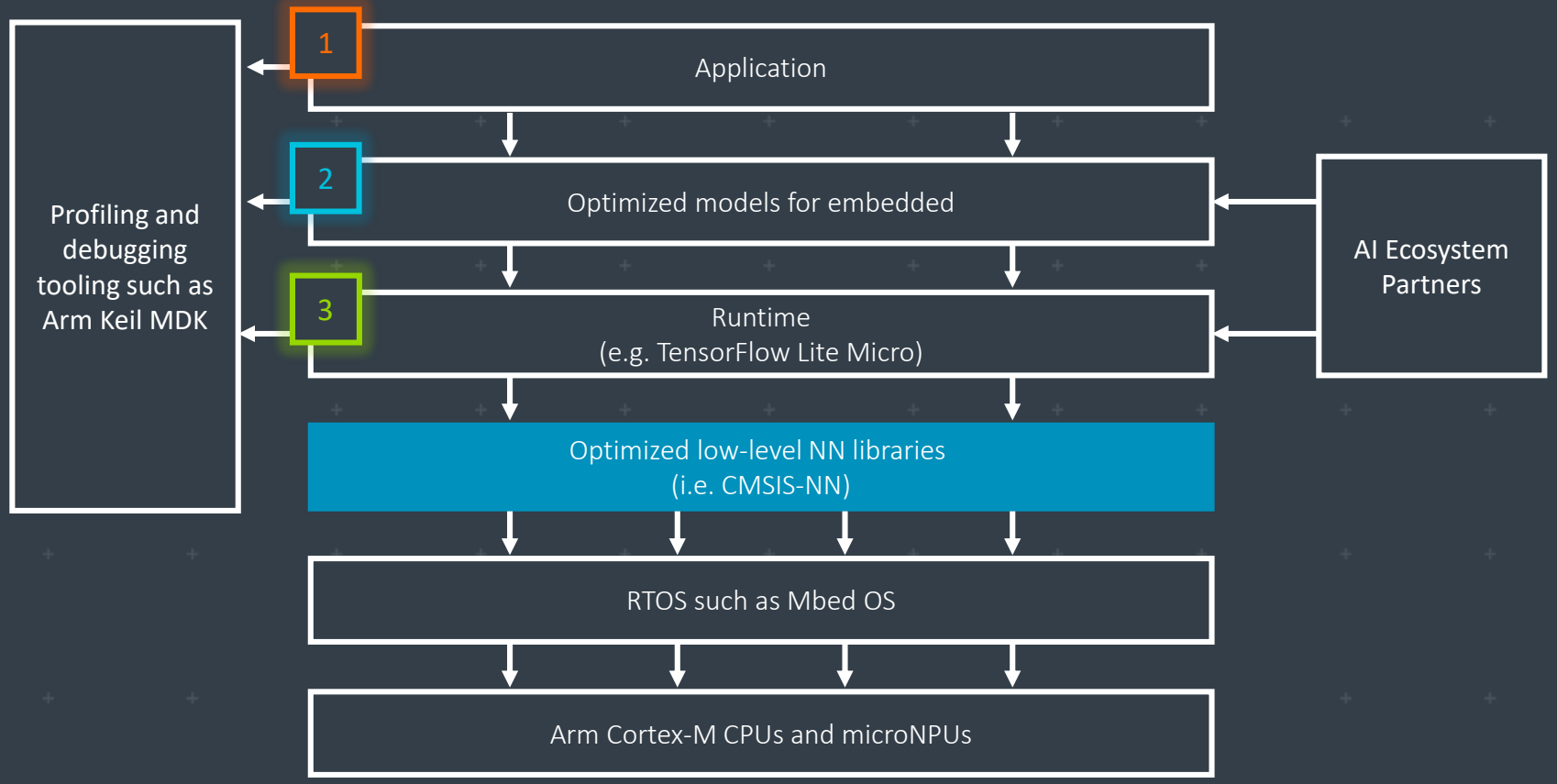
Additional Sponsorships available – contact Olga@tinyML.org for info

Arm: The Software and Hardware Foundation for tinyML

1 Connect to high-level frameworks

2 Supported by end-to-end tooling

3 Connect to Runtime



Stay Connected

@ArmSoftwareDevelopers

@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm

arm

Deeplite

WE USE AI TO MAKE OTHER AI FASTER, SMALLER AND MORE POWER EFFICIENT

Automatically compress SOTA models like MobileNet to <200KB with **little to no drop in accuracy** for inference on resource-limited MCUs

Reduce model optimization trial & error from weeks to days using Deeplite's **design space exploration**

Deploy more models to your device without sacrificing performance or battery life with our **easy-to-use software**

BECOME BETA USER **bit.ly/testdeeplite**

mobilityXlab arm AI 100 CBINSIGHTS

TinyML for all developers

C++ library

Arduino library

WebAssembly

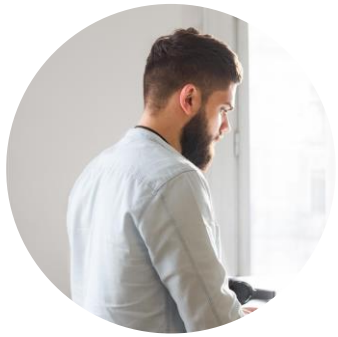
Dataset

Acquire valuable training data securely

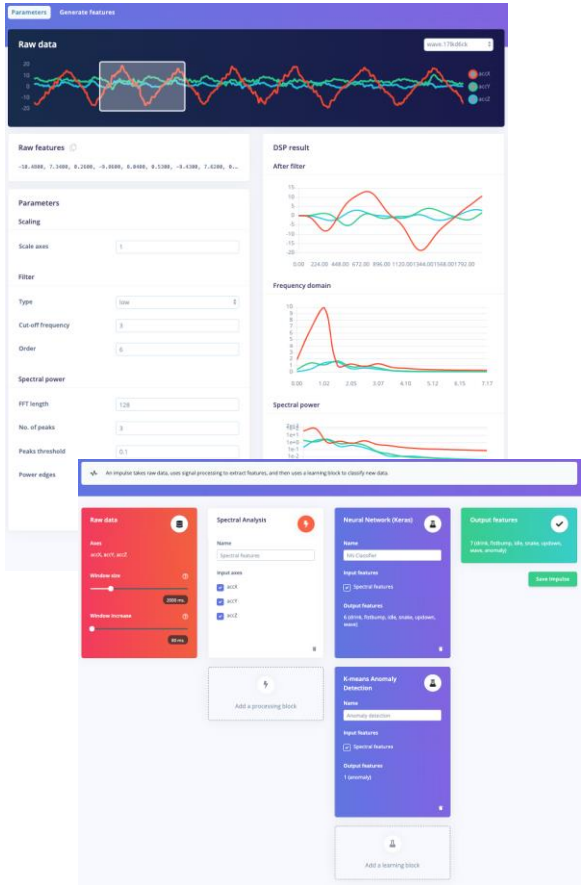
Enrich data and train ML algorithms

Edge Device

Real sensors in real time
Open source SDK



Impulse



Embedded and edge compute deployment options

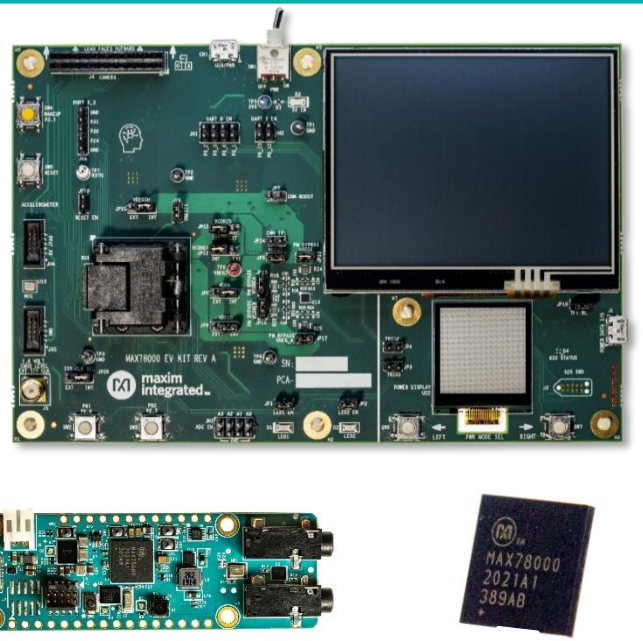
Test impulse with real-time device data flows

Test

www.edgeimpulse.com

Maxim Integrated: Enabling Edge Intelligence

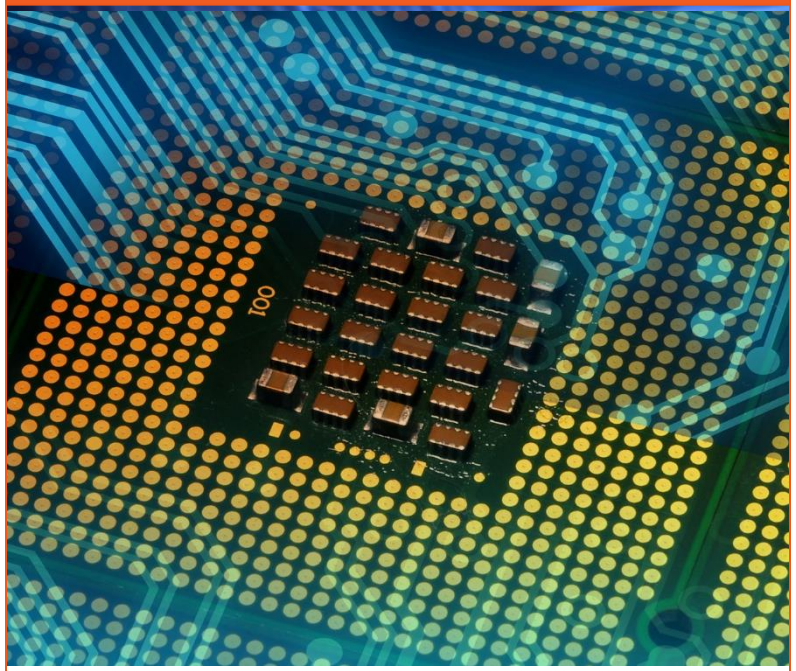
Advanced AI Acceleration IC



The new MAX78000 implements AI inferences at low energy levels, enabling complex audio and video inferencing to run on small batteries. Now the edge can see and hear like never before.

www.maximintegrated.com/MAX78000

Low Power Cortex M4 Micros



Large (3MB flash + 1MB SRAM) and small (256KB flash + 96KB SRAM, 1.6mm x 1.6mm) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

www.maximintegrated.com/microcontrollers

Sensors and Signal Conditioning

Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

www.maximintegrated.com/sensors

maxim integrated™

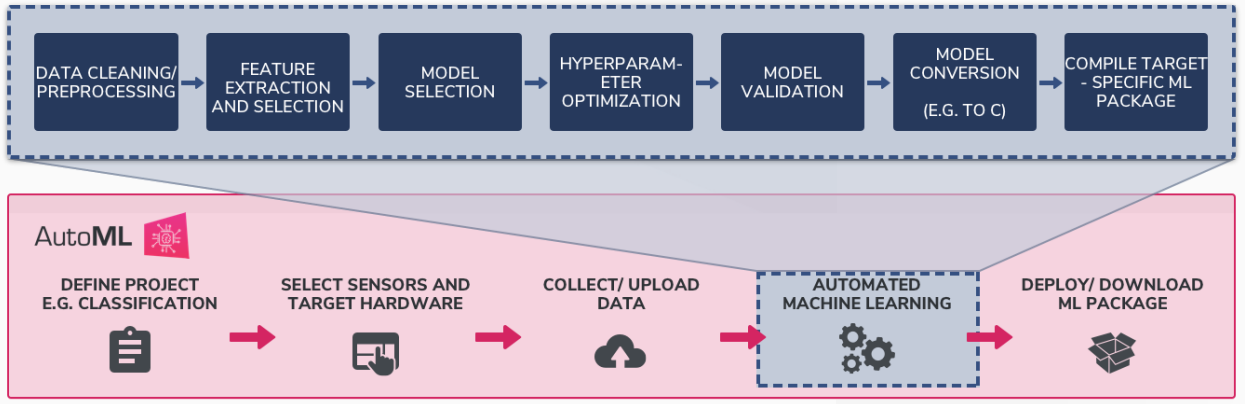
Qeexo AutoML

Automated Machine Learning Platform that builds tinyML solutions for the Edge using sensor data

Key Features

- Supports 17 ML methods:
  - Multi-class algorithms: GBM, XGBoost, Random Forest, Logistic Regression, Gaussian Naive Bayes, Decision Tree, Polynomial SVM, RBF SVM, SVM, CNN, RNN, CRNN, ANN
  - Single-class algorithms: Local Outlier Factor, One Class SVM, One Class Random Forest, Isolation Forest
- Labels, records, validates, and visualizes time-series sensor data
- On-device inference optimized for low latency, low power consumption, and small memory footprint applications
- Supports Arm® Cortex™- M0 to M4 class MCUs

End-to-End Machine Learning Platform



For more information, visit: www.qeexo.com

Target Markets/Applications

- Industrial Predictive Maintenance
- Smart Home
- Wearables
- Automotive
- Mobile
- IoT

Reality AI®

Add Advanced Sensing to your Product with Edge AI / TinyML

https://reality.ai info@reality.ai @SensorAI Reality AI

Pre-built Edge AI sensing modules, plus tools to build your own

Reality AI solutions

Prebuilt sound recognition models for indoor and outdoor use cases

Solution for industrial anomaly detection

Pre-built automotive solution that lets cars "see with sound"

Reality AI Tools® software

Build prototypes, then turn them into real products

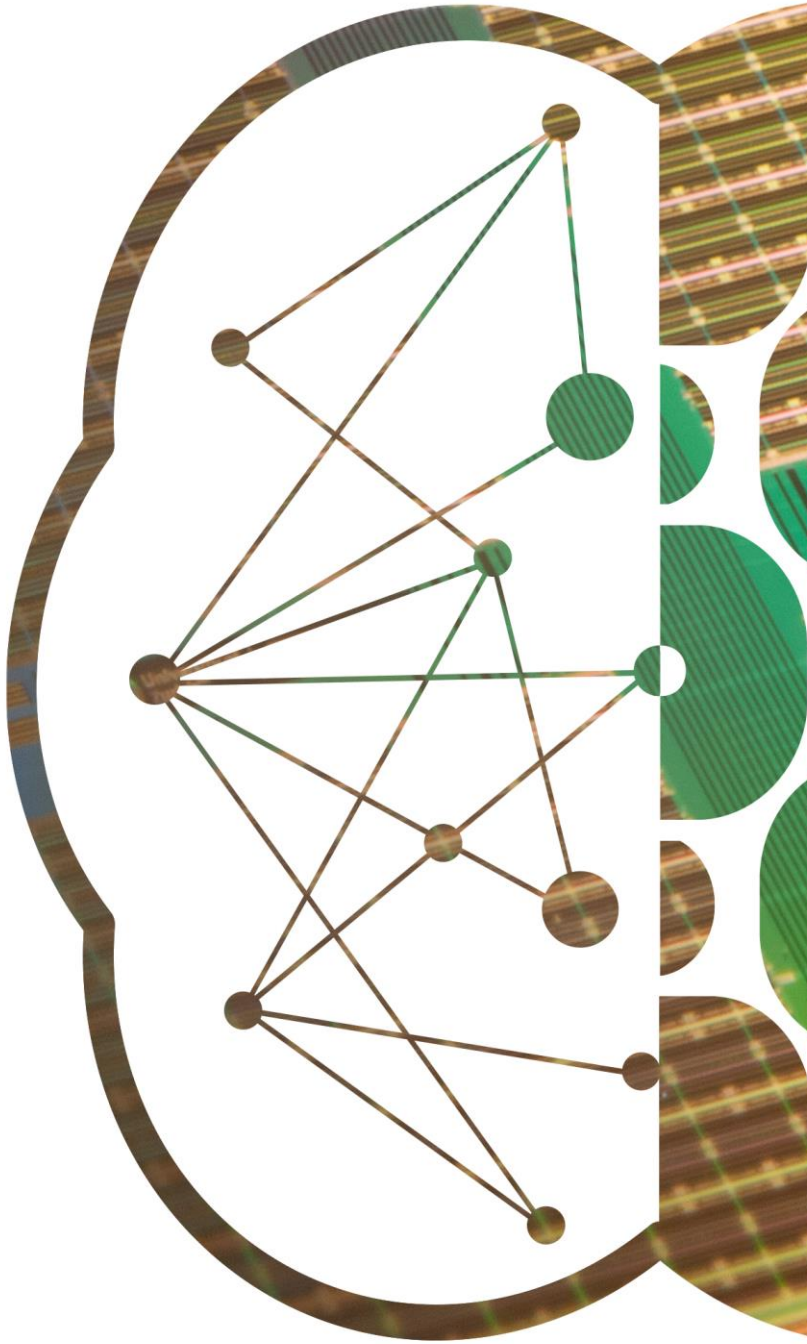
Explain ML models and relate the function to the physics

Optimize the hardware, including sensor selection and placement

SynSense

SynSense builds **sensing and inference** hardware for **ultra-low-power** (sub-mW) **embedded, mobile and edge** devices. We design systems for **real-time always-on smart sensing**, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai



Next tinyML Talks

Date	Presenter	Topic / Title
Tuesday, April 13	**Bernhard Suhm** Machine Learning Product Manager, MathWorks	Deploying AI to Embedded Systems

Webcast start time is 8 am Pacific time

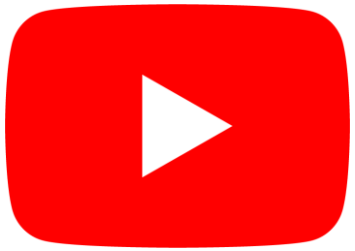
Please contact talks@tinyml.org if you are interested in presenting

TINY ML TALKS webcast

Reminders

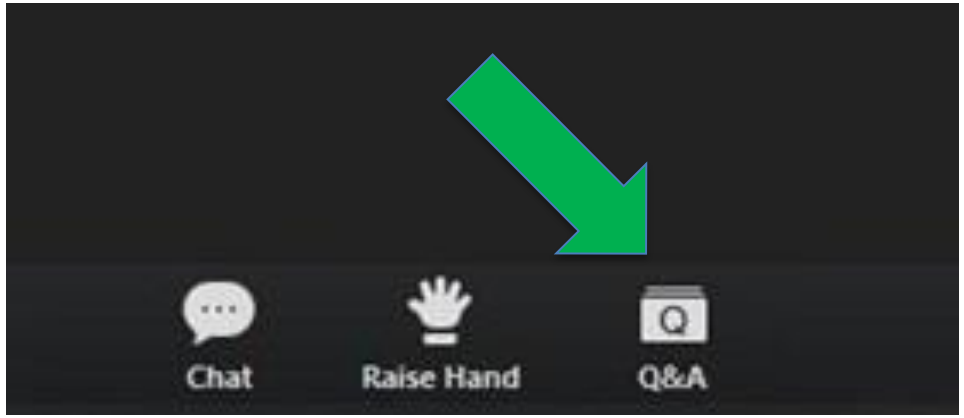
Slides & Videos will be posted tomorrow

tinyml.org/forums



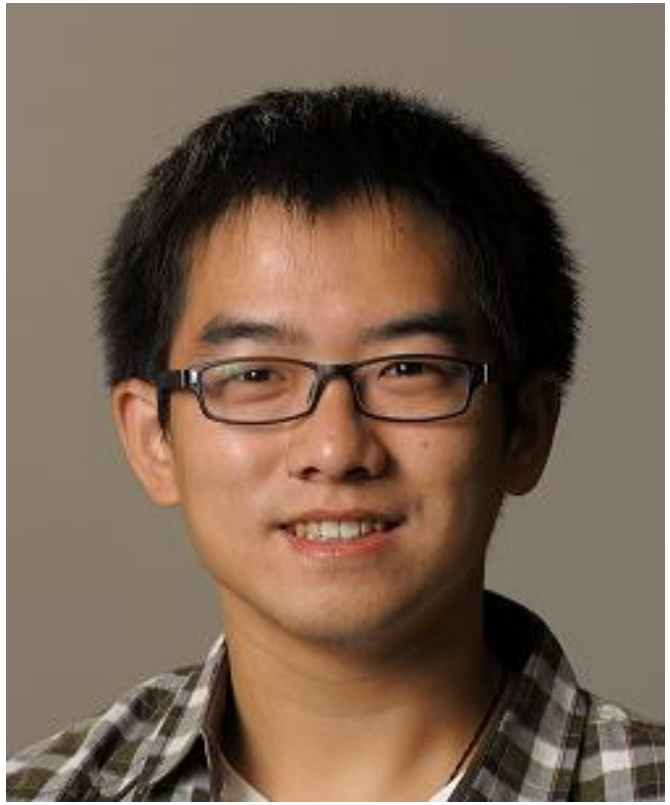
youtube.com/tinyml

Please use the Q&A window for your questions



TINY ML TALKS webcast

Atlas Wang



Professor Atlas Wang is currently an Assistant Professor of Electrical and Computer Engineering at UT Austin, leading the VITA research group (https://vita-group.github.io/). He is broadly interested in the fields of machine learning, computer vision, optimization, and their interdisciplinary applications. His latest interests focus on automated machine learning (AutoML), learning-based optimization, machine learning robustness, and efficient deep learning. He has received many research awards and scholarships, including most recently an ARO Young Investigator award, an IBM Faculty Research Award, an Amazon Research Award (AWS AI), an Adobe Data Science Research Award, and four research competition prizes from CVPR/ICCV/ECCV.

tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

The Lottery Ticket Hypothesis in Gigantic Pre-trained Models: *What, Why and How*

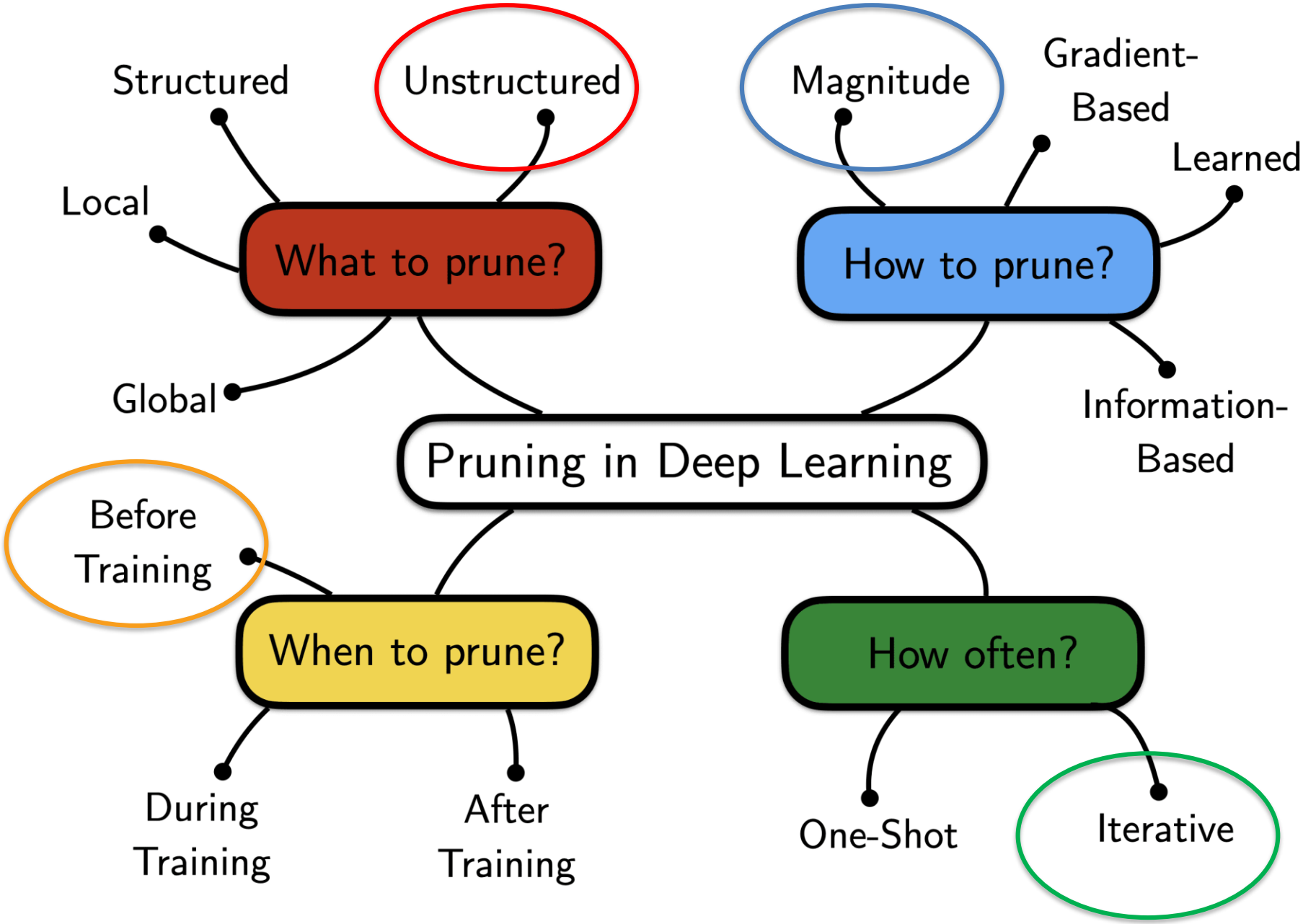
Atlas Wang, ECE@UT Austin

Work in Collaboration with: Tianlong Chen (UT Austin); Jonathan Frankle & Michael Carbin (MIT); Shiyu Chang, Sijia Liu & Yang Zhang (MIT-IBM Watson AI Lab)

www.tinyML.org

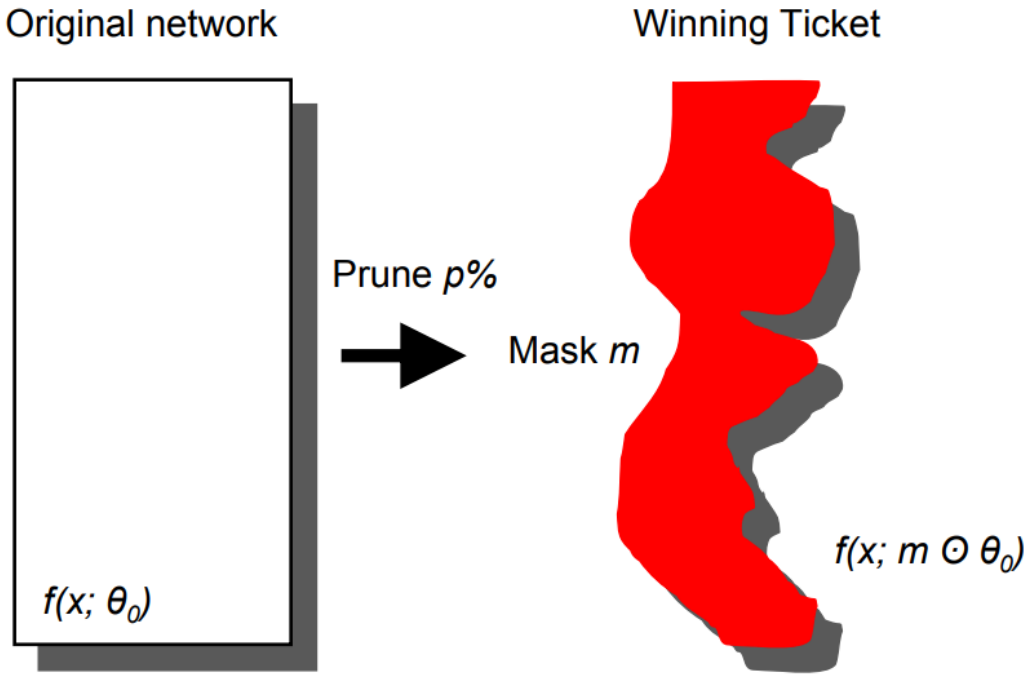
The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

TINY ML TALKS webcast



TINY ML TALKS webcast

The Lottery Ticket Hypothesis. *A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.*

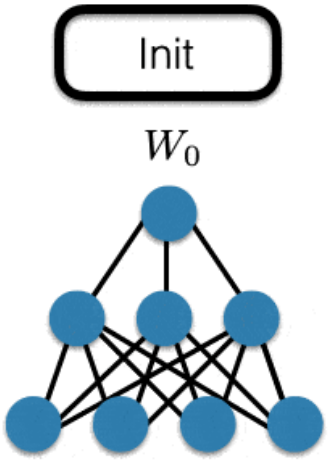


- Winning Ticket gives
  - Better or same results
  - Shorter or same training time
  - Notably fewer parameters
  - Is trainable from the beginning

…As long as we know which sub-network is winning!

TINY ML TALKS webcast

Searching for Tickets: Iterative Magnitude Pruning



Frankle & Carbin, 2019
Viz: @RobertTLange

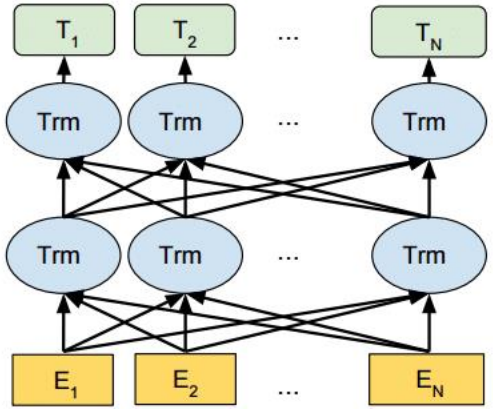
TINY ML TALKS webcast

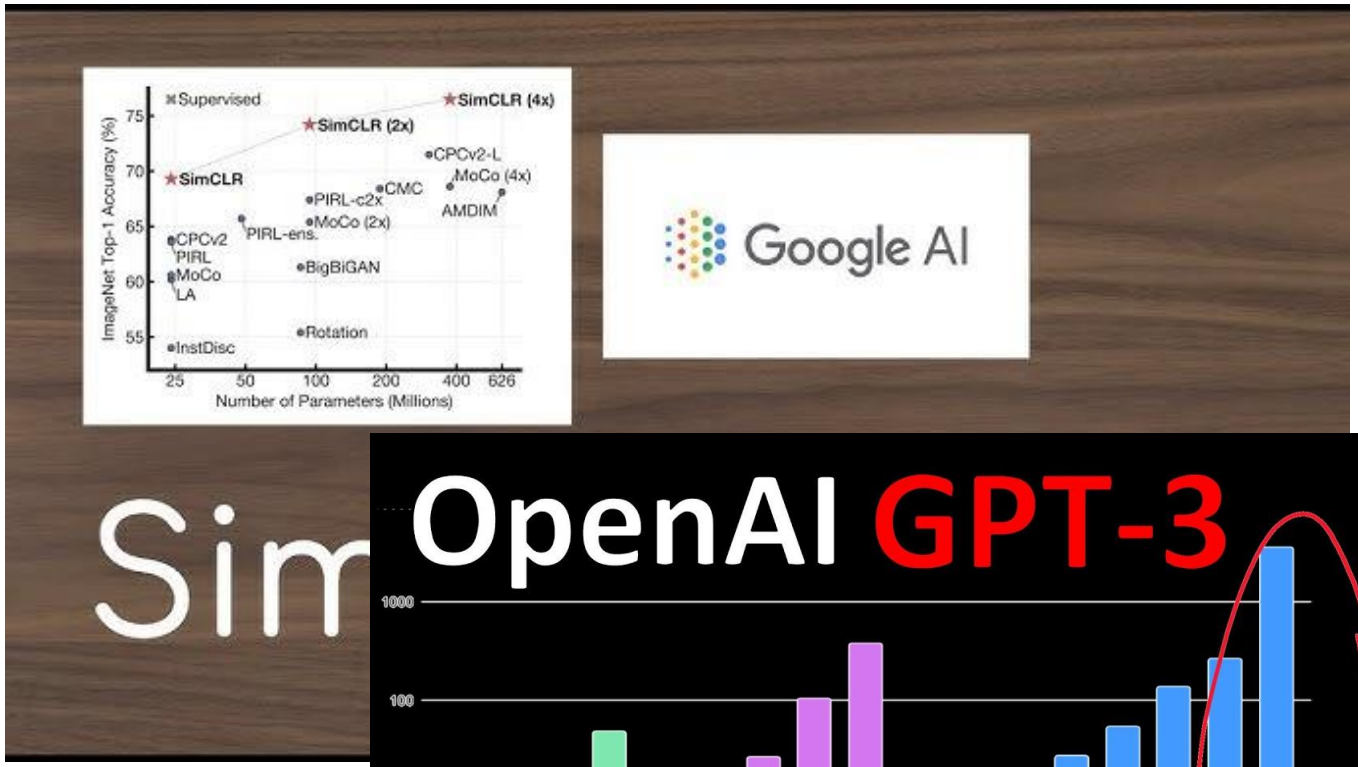
Many efforts follow since then …

- **Stabilizing the Lottery Ticket Hypothesis (2019):** "Rewinding" (no longer random init)
- **Deconstructing lottery tickets: Zeros, Signs, and the Supermask (2019):** "Masking is training, and signs are crucial"
- **One ticket to win them all (2019):** "One could identify a matching ticket on one vision dataset (e.g. ImageNet) and transfer it to another (e.g. CIFAR-100)"
- **Linear Mode Connectivity (2020):** "Good tickets are connected"
- **Drawing Early-Bird Tickets: Towards more efficient training of deep networks (2020):** "Good sparse masks can emerge in early training (but not the init)"
- **The Early Phase of Neural Network Training (2020):** " It appears very hard to avoid rewinding since the emergence of matching initialization appears highly non-trivial"

TINY ML TALKS webcast

We want to draw lottery from: **Large pre-trained models**





- **Why this goal?**

TINY ML TALKS webcast

We want to draw lottery from: **Large pre-trained models**

Why this goal?

- **Methodology:** new questions arise …
  - Might pre-trained weight eliminate "**rewinding**"?
  - Naturally training in "locality"
- **Practice:** pre-training + transfer is becoming the central paradigm of CV, NLP, and more
  - Can a "lottery" **universally** transfer to all downstream tasks the same well?
  - If yes, the extraordinary cost of finding tickets can be amortized by **re-using**

TINY ML TALKS webcast

1st Subject: BERT

Our Findings in A Nutshell:

- Using unstructured iterative magnitude pruning, we find matching subnetworks at between **40%** and **90%** sparsity in BERT models on standard GLUE and SQuAD downstream tasks.
- We find these subnetworks at **pre-trained initialization**, rather after some amount of training. As in previous work, these subnetworks outperform those found by pruning randomly and randomly reinitializing.
- Subnetworks at 70% sparsity found using the masked language modeling task (the task used for BERT pre-training) **are universally transferable to** other tasks while maintaining accuracy.

TINY ML TALKS webcast

The Existence of Matching Subnetworks in BERT

- [Q 1] Are there winning tickets?
- [Q 2] Are IMP winning tickets sparser than randomly pruned or initialized subnetworks?

Table 2: Performance of subnetworks at the highest sparsity for which IMP finds winning tickets on each task. To account for fluctuations, we consider a subnetwork to be a winning ticket if its performance is within one standard deviation of the unpruned BERT model. Entries with errors are the average across five runs, and errors are the standard deviations. IMP = iterative magnitude pruning; RP = randomly pruning; θ_0 = the pre-trained weights; θ_0' = random weights; θ_0'' = randomly shuffled pre-trained weights.

Dataset	MNLI	QQP	STS-B	WNLI	QNLI	MRPC	RTE	SST-2	CoLA	SQuAD	MLM
Sparsity	70%	90%	50%	90%	70%	50%	60%	60%	50%	40%	70%
Full $\text{BERT}_{\text{BASE}}$	82.4 ± 0.5	90.2 ± 0.5	88.4 ± 0.3	54.9 ± 1.2	89.1 ± 1.0	85.2 ± 0.1	66.2 ± 3.6	92.1 ± 0.1	54.5 ± 0.4	88.1 ± 0.6	63.5 ± 0.1
$f(x, m_{\text{IMP}} \odot \theta_0)$	82.6 ± 0.2	90.0 ± 0.2	88.2 ± 0.2	54.9 ± 1.2	88.9 ± 0.4	84.9 ± 0.4	66.0 ± 2.4	91.9 ± 0.5	53.8 ± 0.9	87.7 ± 0.5	63.2 ± 0.3
$f(x, m_{\text{RP}} \odot \theta_0)$	67.5	76.3	21.0	53.5	61.9	69.6	56.0	83.1	9.6	31.8	32.3
$f(x, m_{\text{IMP}} \odot \theta_0')$	61.0	77.0	9.2	53.5	60.5	68.4	54.5	80.2	0.0	18.6	14.4
$f(x, m_{\text{IMP}} \odot \theta_0'')$	70.1	79.2	19.6	53.3	62.0	69.6	52.7	82.6	4.0	24.2	42.3

TINY ML TALKS webcast

The Existence of Matching Subnetworks in BERT

- [Q 3] Does rewinding improve performance?
- [Q 4] Do IMP subnetworks match the performance of standard pruning?

Table 3: Performance of subnetworks found using IMP with rewinding to the steps in the left column and standard pruning (where subnetworks are trained using the final weights from the end of training).

Dataset	MNLI	QQP	STS-B	WNLI	QNLI	MRPC	RTE	SST-2	CoLA	SQuAD	MLM
Sparsity	70%	90%	50%	90%	70%	50%	60%	60%	50%	40%	70%
Full $\text{BERT}_{\text{BASE}}$	82.39	90.19	88.44	54.93	89.14	85.23	66.16	92.12	54.51	88.06	63.48
Rewind 0% (i.e., θ_0)	82.45	89.20	88.12	54.93	88.05	84.07	66.06	91.74	52.05	87.74	63.07
Rewind 5%	82.99	88.98	88.05	54.93	88.85	83.82	62.09	92.43	53.38	87.78	63.18
Rewind 10%	82.93	89.08	88.11	54.93	89.02	84.07	62.09	92.66	52.61	87.77	63.49
Rewind 20%	83.08	89.21	88.28	55.75	88.87	85.78	61.73	92.89	52.02	87.36	63.82
Rewind 50%	82.94	89.54	88.41	53.32	88.72	85.54	62.45	92.66	52.20	87.26	64.21
Standard Pruning	82.11	89.97	88.51	52.82	89.88	85.78	62.95	90.02	52.00	87.12	63.77

Transfer Learning for BERT Winning Tickets

- [Q 5] Do winning tickets transfer across tasks?
- [Q 6] Are there patterns in subnetwork transferability?

Figure 2: Transfer Winning Tickets. The performance of transferring IMP subnetworks between tasks. Each row is a source task $\mathcal{S}$. Each column is a target task $\mathcal{T}$. Each cell is $\text{TRANSFER}(\mathcal{S}, \mathcal{T})$: the performance of finding an IMP subnetwork at 70% sparsity on task $\mathcal{S}$ and training it on task $\mathcal{T}$ (averaged over three runs). **Dark cells mean the IMP subnetwork on task $\mathcal{S}$ is a winning ticket on task $\mathcal{T}$ at 70% sparsity, i.e., $\text{TRANSFER}(\mathcal{S}, \mathcal{T})$ is within one standard deviation of the performance of the full BERT network.** The number on the right is the number of target tasks $\mathcal{T}$ for which transfer performance is at least as high as same-task performance. The last row is the performance when the pruning mask comes from directly pruning the pre-trained weights θ_0.

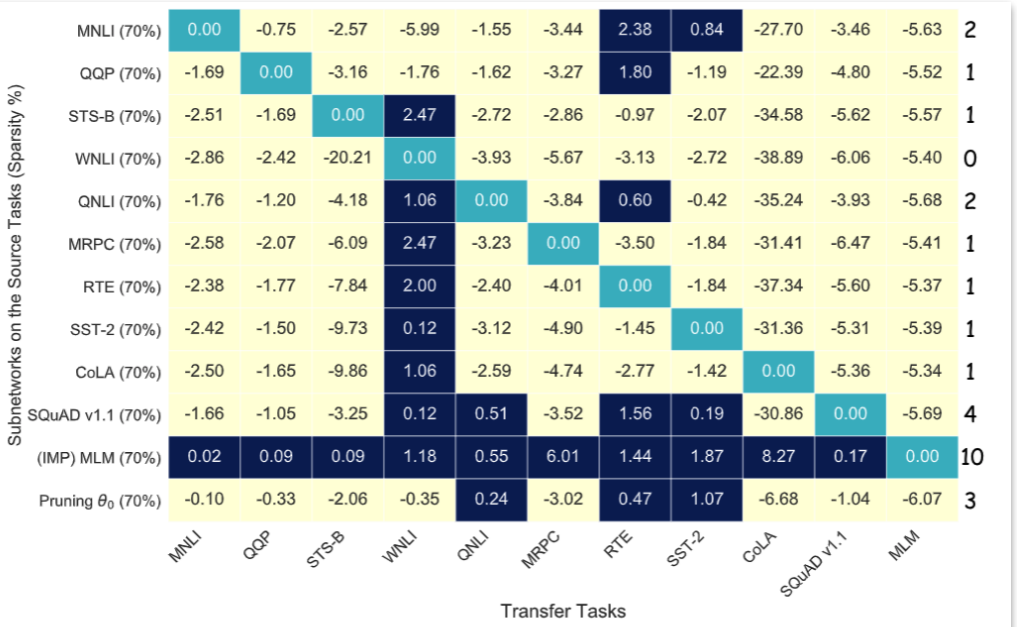
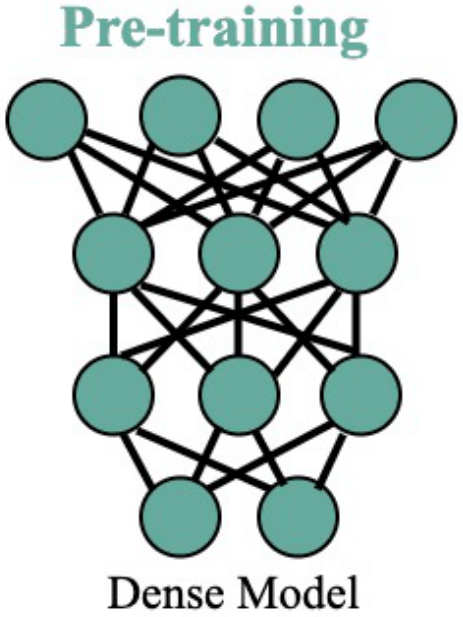


Figure 3: Transfer vs. Same-Task. The performance of transferring IMP subnetworks between tasks. Each row is a source task $\mathcal{S}$. Each column is a target task $\mathcal{T}$. Each cell is $\text{TRANSFER}(\mathcal{S}, \mathcal{T}) - \text{TRANSFER}(\mathcal{T}, \mathcal{T})$: the transfer performance at 70% sparsity minus the same-task performance at 70% sparsity (averaged over three runs). **Dark cells mean the IMP subnetwork found on task $\mathcal{S}$ performs at least as well on task $\mathcal{T}$ as the subnetwork found on task $\mathcal{T}$.**

TINY ML TALKS webcast

2nd Subject: Computer Vision



This is more difficult and subtle than BERT! **Why?**

TINY ML TALKS webcast

2nd Subject: Computer Vision

Our Findings in A Nutshell:

- Using unstructured iterative magnitude pruning, we identify matching sub-networks up to 67.23%, 59.04%, 95.60% sparsity, at pre-trained weights from ImageNet-equipped supervised pre-training, simCLR and MoCo, respectively.
- We also find matching subnetworks at pre-trained initialization with sparsity from 73.79% to 98.20% in a variety of classification, detection and segmentation downstream tasks.
- Subnetworks at 67.23%, 59.04% and 59.04% sparsity, found respectively using supervised ImageNet, simCLR and MoCo pre-training, are **universally transferable** to diverse downstream classification tasks with nearly same accuracies.

TINY ML TALKS webcast

More Properties of CV Pre-training Tickets

Unlike previous matching subnetworks found at random initialization or rewinding, those identified at pre-trained initialization are more sensitive to structure perturbations.

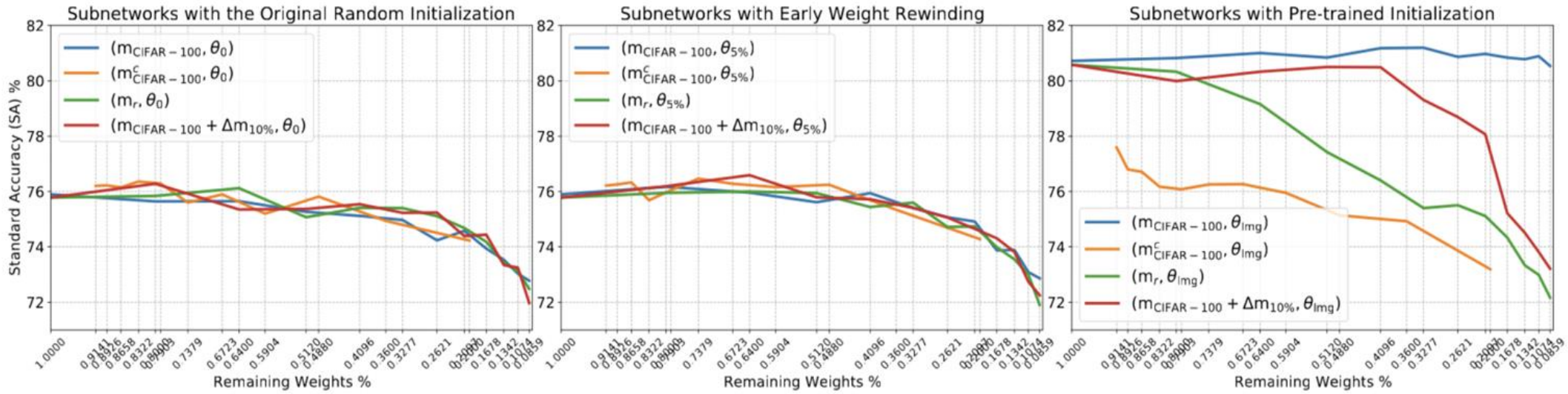


Figure 8. Performance comparison across subnetworks found on CIFAR-100 with the original random initialization θ_0, early rewinding weight $\theta_{5\%}$, and the pre-trained weights θ_{Img}. $m_{\text{CIFAR}-100}$ = masks found by IMP; $m^c_{\text{CIFAR}-100}$ = the complementary masks of $m_{\text{CIFAR}-100}$, where $m \cap m^c = \emptyset$ and $m \cup m^c = \mathbf{1} \in \mathbb{R}^{d_1}$; m_r = random pruned mask; $\Delta m_{10\%}$ = mask perturbations by randomly flipping 10% "1" and 10% "0" in the mask $m \in \{0,1\}^{d_1}$ to its opposite value. Curves are the average across three independent runs.

TINY ML TALKS webcast

More Properties of CV Pre-training Tickets

Different pre-training yield diverse mask structures and perturbation sensitivities.

Figure 6. Top: The relative mask similarity between subnetworks which identified on supervised ImageNet, simCLR and MoCo pre-training tasks. Bottom: The number of completely pruned (zero) kernels in subnetworks found on different pre-training tasks.

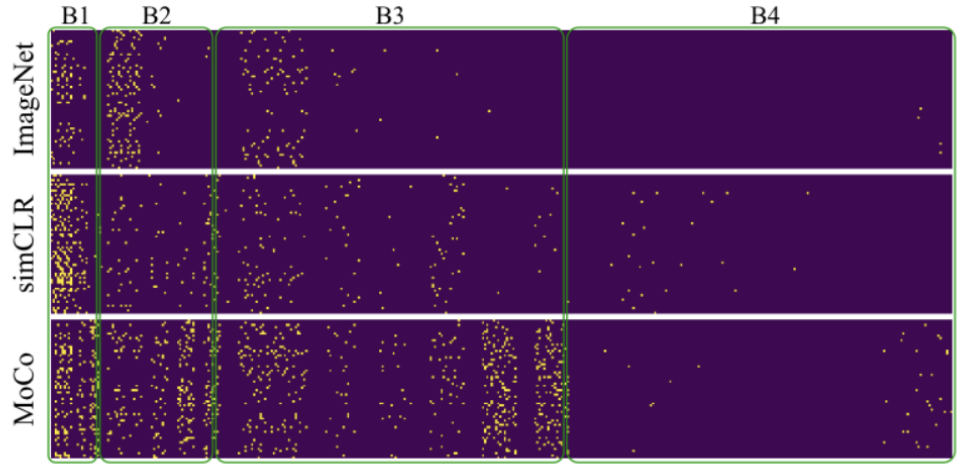


Figure 7. Kernel-wise heatmap visualizations of subnetworks with 79.03% sparsity found on supervised ImageNet, simCLR and MoCo pre-training tasks. From left to right, we visualization all kernels of subnetworks from the input to the output layers. The bright dots (●) represent the completely pruned (zero) kernels and the dark dots (●) the kernels having at least one unpruned weight. B1∼B4 donate four residual blocks in the ResNet-50 backbone.

More Properties of CV Pre-training Tickets

Pruning from larger pre-trained models produces better transferable matching subnetworks

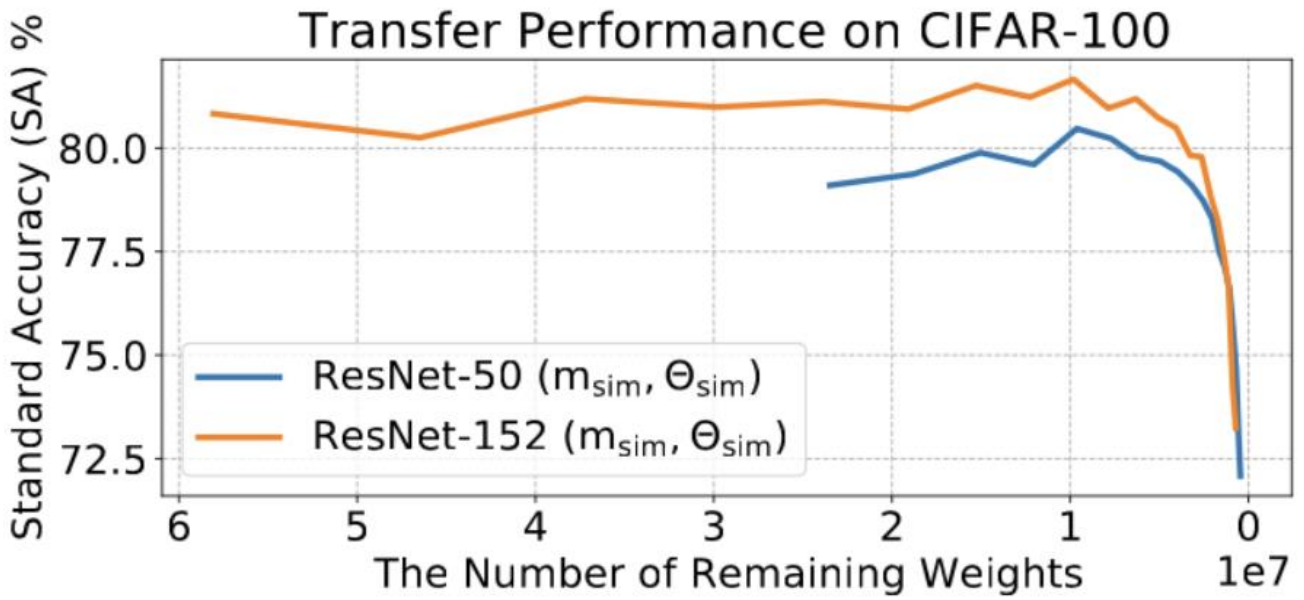


Figure 9. Transfer performance on CIFAR-100 over the number of remaining weights. Subnetworks are found on the simCLR pre-training task with pre-trained ResNet-50 and ResNet-152 weights.

Promise of LTH in “tiny” ML

- Many speech recognition and some simple CV models (like MNIST classification) can already run on MCUs, e.g. SparkFun.
- More re… sks onto M…
- **H…**
  - …ching
  - …
  - …sNet 12 or smaller), …

New Regime:

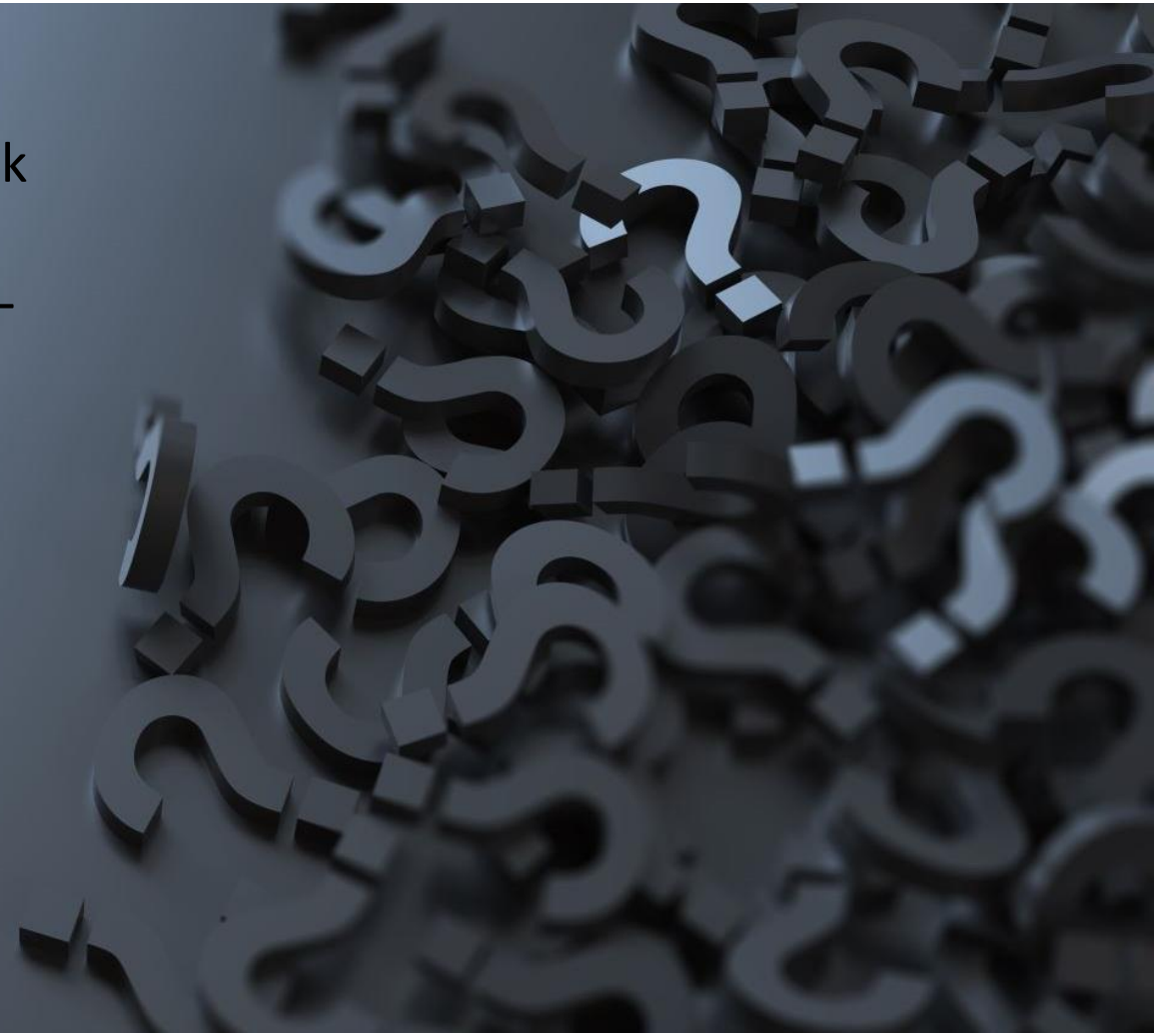
Pre-training a large model

-> finding its lottery

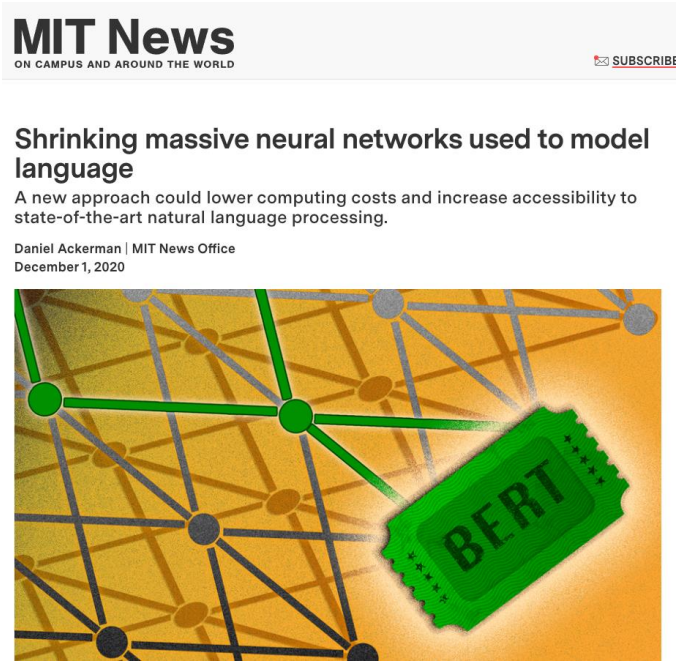
-> deploy and fine-tuning

Practical Relevance of LTH: Translation onto Hardware Savings?

- Extremely high unstructured sparsity can already be accelerated on smartphone processors, e.g., XNNPack
- There is a hope to finding LTH at structured sparsity – we are working on it!
- The benefits of high sparsity extend beyond training and inference efficiency:
  - An effective regularization, for data-efficient training, and for avoiding overfitting/forgetting
  - Sparse models save communication bandwidths too – likely a good friend for federated learning!



TINY ML TALKS webcast



References

- T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, M. Carbin, and Z. Wang, *"The Lottery Tickets Hypothesis for Supervised and Self-supervised Pre-training in Computer Vision Models",* **CVPR 2021**
- T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, Z. Wang, and M. Carbin, "*The Lottery Ticket Hypothesis for Pre-trained BERT Networks",* **NeurIPS 2020.**

Project Webpage: https://tianlong-chen.github.io/Project_Page/index.html

… And All Our Research Codes: https://github.com/VITA-Group

Copyright Notice

www.tinyML.org