Course 1, 2 & 3 Review

Throughout this course, we will continue to use the terminology developed in prior courses. For **returning students**, this will be a recap of the basic language we have been using. For **new students (welcome!)**, this will be an introduction to some of the fundamental parlance used within the machine learning community, as well as an overview of topics studied in courses 1-3.

At a high level, during the previous courses, we have covered the basics of both ML and TinyML, including their lifecycle, learning procedure, and examples of end-to-end deployment through case studies utilizing image-based data, audio data, and sensor data. These examples incorporate all stages of the developmental workflow, including data collection, model training, and model deployment onto a device.

Course 1 Recap (Fundamentals of TinyML)

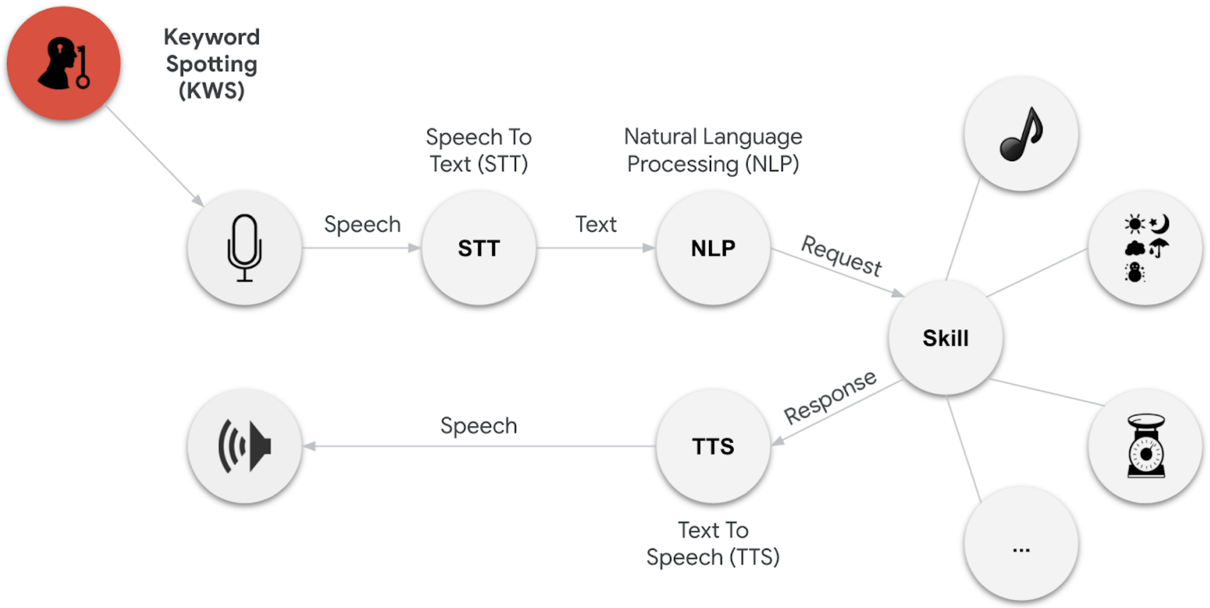
We began this journey by starting from a blank slate. First, we illustrated the potential for TinyML as a game-changing technology, before then highlighting the challenges associated with its development and incorporation into commercial and consumer applications. Following this, we took a deep dive into machine learning. We learned that machine learning is inherently **data-driven**, and thus requires high-quality data (and usually lots of it) in order to learn associations. We learned that in **supervised learning**, we can use feature information (our X variable) in order to predict our output (our Y variable), and can use machine learning to "learn" the mapping between these two sets of variables, using a **training set**. Once we have developed this mapping, we can assess how well our mapping performs by comparing the performance of our predicted output on an unseen set of data, which we call the **test set**. Using this procedure, we can perform both regression and classification tasks.

Following this, we took an even deeper dive into the subfields of machine learning focused on neural networks, and, more specifically, deep learning. We learned that neural networks (NNs) are **universal function approximators** that allow us to approximate any non-linear function by learning from training data. Through a procedure known as backpropagation, we are able to update the weights in our NN to values that more closely match the function we are trying to approximate. This procedure is what is known as "learning" for a NN, and the rate of learning is controlled by a quantity known as a learning rate. NNs can handle any kind of input data, including image, audio, and sensory data. We then delved into convolutional neural networks (CNNs), which work much the same as traditional NNs but are specially designed to accommodate spatial correlations within data. This capability is especially important for images and provides superior performance compared to traditional NNs.

In the final parts of course 1, we saw examples of CNNs for classifying images, discussed TF Data as a method of streaming data, delved into augmentation and regularization techniques to combat overfitting in NNs, and had our first introduction to responsible AI development.

Course 2 Recap (Applications of TinyML)

While course 1 covered the fundamentals of machine learning, course 2 is where our journey into the world of TinyML really began. We started to look at the lightweight version of Tensorflow that Google has designed for microcontrollers, TF Lite, and how to convert our previously developed models into TF Lite versions that could be deployed on microcontrollers. We learned about quantization techniques that can be used for post-training quantization, different optimizations to consider during the conversion process, as well as more advanced techniques such as quantization-aware training. We will recap some of those concepts here as well, but only as they pertain to introducing you to new ideas that are specifically relevant to MLOps.



This culminated in several end-to-end implementations, the first involving **keyword spotting (KWS)**. This application has become increasingly important in commercial applications, and is used in many smart devices for wake word detection, such as Apple's "Hey, Siri" and Google's "OK Google". In our implementation, we taught a microcontroller to be able to respond to a small subset of words, such as "Yes" and "No", by training it on a set of audio data. This use case introduced us to many of the difficulties associated with TinyML, such as the importance of data quality, the difficulty of debugging, and the importance of non-functional requirements such as inference time or model size. We were also introduced to the prospect of generating our own dataset, which is fraught with its own difficulties such as producing sufficient variation in the data to prevent overfitting and ensure good performance of the algorithm during inference.

Our next use case introduced **visual wake word (VWW) detection**: looking at an image to see if a given object is present. This is an important task with many real-world applications, such as person detection, mask detection (particularly prescient since the COVID-19 pandemic), animal detection, and so on. This use case provided us with a nice segue into image data, which is generally more complex to process and more computationally intensive than other forms of data, making it particularly challenging for resource-constrained devices like microcontrollers. To help build CNNs capable of working on microcontrollers, we were introduced to lightweight architectures suchas MobileNet, which utilize depthwise separable convolutions to minimize the number of computations per image. **Transfer learning** was also discussed, wherein high-performance networks such as VGG-16 and Inception can be adapted to the task at hand in circumstances where the original task is similar to the current task. Anomaly detection was also discussed as a separate use case that touches on the paradigm of unsupervised learning.

These use cases helped us to gain a more visceral feel of the ML workflow, going through the stages of data collection, data preparation, model design, model training, model optimization, and model conversion. We left the final stages, model deployment and model inference, for course 3, since this requires a microcontroller to deploy our models on!

Course 3 Recap

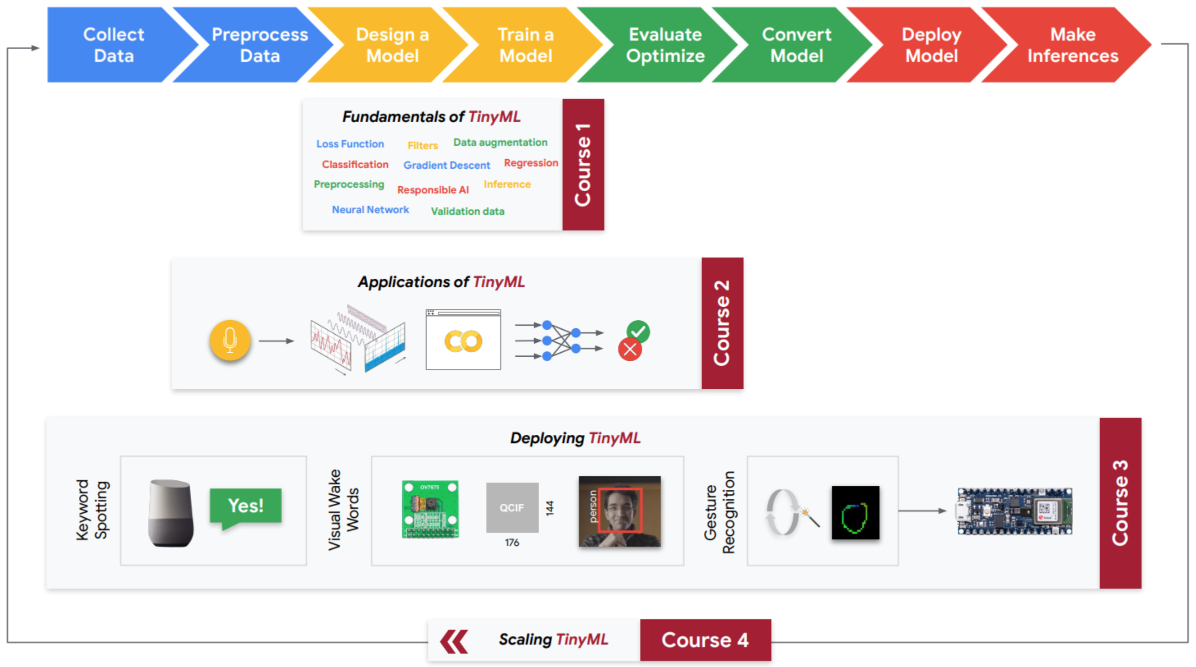
Course 3 is where all the fun began! Not only did we now have our models at the ready, but in this course we learned how to deploy them on a Nano 33 BLE Sense from our TinyML Kit and tick off the last two sections of the ML workflow: model deployment, and model inference.

This course is where we really delved into TF Lite, and its even lighter framework, TF Lite Micro. We also introduced to hardware-specific terminology, such as Arduino "cores", embedded frameworks, mbedOS, and bare metal machines, and how each of these technologies are relevant to our TinyML applications. We spent time discussing the memory hierarchy and the trade-offs between memory size, cost, and read time, and how this can influence the performance of our system. Following this, we discussed low-level aspects of the TF Lite Micro, such as the FlatBuffer files used for model storage, the pre-allocated Tensor Arena used for performing intermediate calculations, and OpsResolver to import only the operations that are needed by our given model. After covering these essentials, we moved onto some end-to-end implementations of the projects covered previously, beginning with keyword spotting, then visual wake word detection, and finally the development of a gesturing magic wand that relies on accelerometer data embedded within our Nano 33 BLE Sense.

What's Next?

Finally throughout Courses 1, 2, and 3 we explored Tiny Machine Learning through the lens of the typical end-to-end workflow from Data Collection to Training to Deployment to better understand all of the steps needed to deploy a successful TinyML application. Through MLOps we will expand beyond the workflow discussed in the previous courses and consider all of the things one would need to do to scale their models and optimize their performance.

MLOps is an important aspect of machine learning that is often overlooked, but is one of the most essential skills for developing robust applications for commercial and consumer applications. Hopefully, you are familiar with all of the review material we have discussed. If not, we suggest you go back and look at some of the previous courses to help jog your memory.


Collect Data
Preprocess Data
Design a Model
Train a Model
Evaluate Optimize
Convert Model
Deploy Model
Make Inferences
Fundamentals of TinyML
Loss Function
Filters
Data augmentation
Classification
Gradient Descent
Regression
Preprocessing
Responsible AI
Inference
Neural Network
Validation data
Course 1
Applications of TinyML
Course 2
Deploying TinyML
Keyword Spotting
Yes!
Visual Wake Words
QCIF
144
176
person
Gesture Recognition
Course 3
Scaling TinyML
Course 4