EDUCATION & TECHNICAL ENGINEERING

Deep Learning and Signal Processing: A Comprehensive Guide to CS7643 and ECE3084 at Georgia Institute of Technology

Published: January 11, 2025 | Index ID: #210

The convergence of computational power and algorithmic innovation has positioned the **Georgia Institute of Technology (Georgia Tech)** as a global leader in the fields of machine learning and electrical engineering. Specifically, through its Online Master of Science in Computer Science (OMSCS) and its School of Electrical and Computer Engineering (ECE), Georgia Tech provides a rigorous academic framework for understanding the transition from classical signal processing to advanced neural architectures. Two cornerstone courses—**CS7643: Deep Learning** and **ECE3084: Signals and Systems**—represent the theoretical and practical spectrum required to master modern artificial intelligence.

The Academic Framework of Georgia Tech's Technical Curriculum

Georgia Tech is consistently ranked as a top-tier public university, recently recognized by the **Princeton Review** as the No. 1 Best Value Public University. This reputation is built upon a foundation of academic rigor and accessibility, particularly through the **OMSCS program**, which delivers high-level computer science education via 100% web-based platforms like Canvas. The curriculum for courses such as CS7643 is designed to be exhaustive, covering everything from the fundamental linear algebra of neural networks to the complex implementation of **Graph Neural Networks (GNNs)**.

The institutional focus remains on bridging the gap between theoretical research and industry application. For graduate students, this often involves acting as **Graduate Teaching Assistants (GTAs)**, facilitating the learning process for thousands of students globally while pursuing dual degrees in disciplines like **Electrical and Computer Engineering (ECE)**and Business Administration (MBA). This multidisciplinary approach ensures that graduates are not just coders, but architects of complex systems.

Fundamental Foundations: From ECE3084 to Deep Learning

Before diving into the complexities of deep neural networks, it is essential to understand the mathematical bedrock provided by **ECE3084: Signals and Systems**. This course introduces the concepts of **Continuous-Time (CT)** and **Discrete-Time (DT)**signals, which are the precursor to the data structures used in machine learning. Signals are essentially functions that convey information, and systems are the mathematical operators that transform these signals.

Key Concepts in Signals and Systems

- Linear Time-Invariant (LTI) Systems: The core of classical signal processing, where the output is determined by the convolution of the input signal and the system's impulse response.
- Frequency Response: Understanding how systems behave across different frequencies, which is vital for filtering noise in data—a common step in preprocessing for Deep Learning.
- Transfer Functions: Mathematical representations in the s-domain or z-domain that describe the relationship between input and output.

In the context of **CS7643**, these concepts evolve. A neural network can be viewed as a highly non-linear, adaptive system. While ECE3084 focuses on predictable, linear transformations, CS7643 explores how high-dimensional data can be transformed through successive layers of weights and non-linear activation functions to perform classification or regression tasks.

Technical Deep Dive: The Mechanics of 2D Convolution

Convolution is perhaps the most critical operation in modern computer vision. In **CS7643 Problem Set 3 (ps3.pdf)**, students are often tasked with implementing convolution layers from scratch. Unlike the 1D convolution found in audio processing, **2D Convolution** is used to extract spatial features from images.

The Mathematical Model of 2D Convolution

The 2D convolution of an input image *I* with a kernel *K* is defined by the following discrete sum:

S(i, j) = (I * K)(i, j) = :3£ I(i+m, j+n)K(m, n)

Where:

- I is the input matrix (image).
- K is the kernel or filter matrix (e.g., a 3x3 or 5x5 grid).
- S is the resulting feature map.
- m, n are the indices of the kernel.

Implementation Parameters: Stride and Padding

When implementing test_conv.py as seen in Georgia Tech lab environments, three primary hyperparameters govern the output dimensionality:

1. Kernel Size: The spatial extent of the filter. Common sizes include 3x3 or 5x5.
2. Stride: The number of pixels the kernel moves across the input image. A stride of 2 effectively downsamples the image.
3. Zero Padding: Adding layers of zeros around the border of the input image. This is crucial for maintaining the spatial dimensions of the output and ensuring that the pixels at the edges are processed equally to the pixels in the center.

Parameter	Impact on Output Size	Technical Purpose
Stride (S)	Increases reduction	Reduces computational load and spatial redundancy.
Zero Padding (P)	Increases output size	Prevents information loss at the borders of the image.
Kernel Size (F)	Decreases output size	Determines the receptive field of the neuron.

The formula for calculating the output dimension (O) given an input size (W), filter size (F), padding (P), and stride (S) is:

O = [(W - F + 2P) / S] + 1

Advanced Architectures: DNNs and GNNs

In **L7: Neural Network 101**, the curriculum transitions from basic perceptrons to **Deep Neural Networks (DNNs)** and **Graph Neural Networks (GNNs)**. While DNNs operate on Euclidean data (like grids of pixels or sequences of text), GNNs are designed to operate on non-Euclidean data structures—graphs.

Deep Neural Networks (DNNs)

DNNs utilize multiple hidden layers to learn hierarchical representations. In the context of the **OMSCS CS7643 syllabus**, this involves mastering backpropagation, gradient descent, and regularization techniques like Dropout or Batch Normalization. The goal is to minimize a loss function (e.g., Cross-Entropy) by iteratively updating weights based on the chain rule of calculus.

Graph Neural Networks (GNNs)

GNNs represent a newer frontier. Instead of a fixed grid, the data is represented as **nodes** and **edges**. This is particularly useful in social network analysis, molecular chemistry, and recommendation engines. The core operation in a GNN is **Message Passing**, where each node aggregates features from its neighbors to update its own state. This allows the network to learn the structural topology of the data alongside the feature representations.

Long-Term Memory in Neural Networks

A recurring question in the **Sharc-lab** research at Georgia Tech is: *"How about long-term memory?"* Standard feed-forward networks lack the ability to retain information over time. This leads to the implementation of **Recurrent Neural Networks (RNNs)** and specifically **Long Short-Term Memory (LSTM)**cells. LSTMs solve the vanishing gradient problem by using internal "gates" (input, forget, and output gates) to regulate the flow of information, allowing the network to remember patterns over hundreds of time steps.

Practical Implementation: Python-Based Testing and Verification

Technical proficiency at Georgia Tech is demonstrated through rigorous testing. The file test_conv.py mentioned in the dataset is a testament to the importance of unit testing in deep learning development. When building a **Convolutional Neural Network (CNN)**, one must verify that the forward and backward passes are mathematically consistent.

A Typical Technical Workflow for CNN Testing:

1. Forward Pass Verification: Compare the output of a custom-written convolution function against a known standard (like torch.nn.functional.conv2d).
2. Gradient Checking (Numerical Differentiation): Calculate the gradients of the weights using small perturbations (epsilon) and compare them to the gradients produced by the analytical backpropagation implementation.
3. Boundary Condition Analysis: Testing how the implementation handles various padding sizes and non-square kernels.

Case Study: Problem Set 3 (CS7643)

In **PS3**, students are typically required to implement a modular deep learning library. This involves creating separate classes for Linear layers, ReLU activations, and Convolution layers. The **zero-padding size 1** implementation is a standard requirement, ensuring that for a 3x3 kernel, the spatial dimensions remain constant (if stride=1). This task reinforces the understanding of memory allocation and pointer arithmetic in deep learning frameworks like PyTorch or Caffe.

Comparative Analysis of Technical Paradigms

To better understand the educational path at Georgia Tech, we can compare the focal points of the different curricula mentioned in the research data.

Feature	ECE3084 (Signals & Systems)	CS7643 (Deep Learning)
Primary Data Type	1D Signals (Continuous/Discrete)	High-dimensional Tensors
Key Mathematical Tool	Fourier & Laplace Transforms	Multivariate Calculus & Linear Algebra
System Behavior	Fixed, Linear, Deterministic	Adaptive, Non-linear, Stochastic
Core Operation	Analytical Convolution	Learned Convolution (Kernels as Weights)
Application	Circuit Design, Audio Processing	Image Recognition, NLP, Robotics

Operational Challenges and Troubleshooting in Deep Learning

Even with advanced education, engineers face significant challenges when deploying deep learning models. Georgia Tech's curriculum addresses these through practical lab work.

Common Failure Modes:

- Vanishing/Exploding Gradients: In deep networks, gradients can become infinitesimally small or infinitely large during backpropagation. Solution: Implement Xavier/He initialization or use Batch Normalization layers.
- Overfitting: When the model performs perfectly on training data but fails on unseen data. Solution: Use L2 Regularization (Weight Decay), Dropout, or Data Augmentation.
- Computational Bottlenecks: Large 2D convolutions are expensive. Solution: Transition to Depthwise Separable Convolutions or utilize GPU acceleration via CUDA.

The Role of Graduate Teaching Assistants (GTAs)

As seen with individuals like **Sagar Badlani**, GTAs at Georgia Tech play a vital role in the ecosystem. They bridge the gap between the professor’s theoretical lectures and the students’ coding struggles. Their involvement ensures that the 100% web-based delivery of OMSCS remains high-touch and technically rigorous, providing real-time feedback on complex problem sets like those involving 2D convolution and DNN optimization.

Future Implications: The Evolution of Computational Intelligence

The integration of signals and systems with deep learning is not just an academic exercise; it is the future of engineering. As we move toward **Autonomous Systems** and **Edge AI**, the ability to process signals efficiently (ECE3084) while extracting complex features (CS7643) is paramount. Research at **Sharc-lab** and other Georgia Tech facilities continues to push the boundaries of how neural networks can learn with memory, how they can be optimized for hardware (FPGA/ASIC), and how they can be applied to graph-based data.

Students graduating from these programs are equipped with a dual-threat skill set: the mathematical precision of an electrical engineer and the algorithmic creativity of a computer scientist. This combination is precisely what is required to solve the next generation of global challenges, from climate modeling to personalized medicine. The rigor of the CS7643 syllabus and the foundational clarity of ECE3084 ensure that Georgia Tech remains at the center of this technological revolution.

Ultimately, the journey through Georgia Tech’s technical curriculum is one of increasing abstraction—starting from the physical reality of signals and ending with the cognitive potential of deep neural networks. By mastering the

mechanics of 2D convolution, the architecture of GNNs, and the nuances of system stability, engineers are prepared to build the intelligent systems of tomorrow.

Tags: #Georgia Tech #Deep Learning #OMSCS #Signal Processing #Neural Networks #CS7643 #ECE3084