Great choice! The third idea — **"Predicting Gene Expression from Promoter Sequences"** — is a fascinating mix of genomics, deep learning, and sequence modeling. It touches on key regulatory mechanisms and gives you hands-on experience with biologically meaningful data.

## **🧬 Project Idea: Predicting Gene Expression from Promoter Sequences**

### **🔍 Objective**

To develop a machine learning model, particularly a **deep learning model**, that can predict **gene expression levels** based solely on the **DNA sequence of promoter regions**. The goal is to explore how much of the gene’s expression pattern is encoded in its upstream regulatory DNA sequence.

### **🧠 Biological Rationale**

Promoter regions (~1000 bp upstream of the transcription start site) contain **cis-regulatory elements**—short DNA motifs bound by transcription factors. These motifs influence the recruitment of RNA polymerase and transcription initiation, thus playing a key role in determining how strongly a gene is expressed.

So the hypothesis is:

“The promoter sequence contains enough information to at least partially predict the expression level of a gene.”

### **🧰 What You'll Need**

#### **Data Sources**

* **Promoter sequences:**
  + Ensembl or UCSC Table Browser — extract ~1kb upstream regions from a set of genes.
* **Gene expression data:**
  + **ENCODE**, **GEO (Gene Expression Omnibus)**, or **GTEx** — tissue- or cell line-specific expression values, often in TPM or RPKM.

Make sure your promoter sequences match the organism and tissue of the expression data!

### **🧪 Methodology Overview**

#### **1. Preprocessing**

* Retrieve DNA sequences of promoter regions (e.g., −1000 bp to TSS).
* Normalize expression data (log-transform, scaling).
* Encode DNA sequences (e.g., one-hot encoding, k-mer counts, or embeddings).

#### **2. Model Architecture**

* **Input:** DNA sequence (e.g., 1000 bp, one-hot encoded as 1000×4 matrix).
* **Model types:**
  + 🧠 **Convolutional Neural Networks (CNNs)** – detect local motifs.
  + 🧬 **Recurrent Neural Networks (RNNs)/LSTMs** – capture long-range dependencies.
  + ⚡️ **Transformers** – optional, more advanced, but promising for sequence data.

#### **3. Training**

* **Task:** Regression — predict a continuous gene expression value.
* **Loss function:** Mean Squared Error (MSE).
* **Evaluation metrics:** Pearson/Spearman correlation, R² score.

#### **4. Validation**

* Use **cross-validation** or **train/val/test** split by chromosome (to prevent sequence similarity leakage).
* Plot predicted vs. actual expression levels for visual inspection.

### **🌟 Possible Enhancements**

* **Motif interpretation:** Use saliency maps or integrated gradients to see what parts of the sequence the model uses most — potentially uncover novel motifs!
* **Multi-tissue extension:** Make it a multi-output model to predict expression in several tissues.
* **Add epigenomic features:** Combine sequence with histone marks or DNase-seq data for a multimodal model.

### **🧠 Skills You’ll Gain**

* Biological sequence analysis
* Deep learning for biological data
* Feature encoding for DNA
* Data wrangling from genomic databases
* Model interpretability and visualization

### **🎯 Why This is a Great Undergraduate Project**

* **Feasible**: Can be done on a laptop with a manageable dataset.
* **Educational**: Covers core ideas in bioinformatics and machine learning.
* **Cool Factor**: Gives insights into gene regulation using only raw DNA—like peering into nature’s blueprint with neural nets!

Would you like a more detailed implementation plan? I can help outline the steps, suggest Python libraries (like PyTorch, Biopython, etc.), or even suggest paper references if you want to go deeper. Just let me know! 😊

Awesome! If you’ve got access to **supercomputing resources** and **2 weeks** to work with, you’re in a great position to pull off a **solid, meaningful version** of this project — especially if you scope it well and focus on getting a functional pipeline early.

Let’s break it down:

## **⏳ Is 2 Weeks Enough?**

✅ **Yes**, with focused effort and good planning, **2 weeks is enough** for a **proof-of-concept** that:

* Trains a deep learning model (e.g., CNN or hybrid CNN-RNN)
* Uses real promoter sequences + gene expression data
* Achieves measurable predictive performance (e.g., correlation between predicted and actual expression)
* Has some basic interpretability (e.g., saliency maps or attention weights)

But you’ll want to **prioritize core functionality** over polishing everything.

## **📅 Suggested 2-Week Plan (High-Level)**

### **🗓️ Days 1–2: Data Collection & Preprocessing**

* Select 1 organism (e.g., *Homo sapiens* or *Mus musculus*).
* Download:  
  + Promoter sequences (Ensembl, UCSC, or BioMart).
  + Expression data (ENCODE, GTEx, or GEO).
* Write a script to extract −1000 to TSS regions.
* One-hot encode or convert sequences into k-mer counts (start simple).

### **🗓️ Days 3–5: Build Baseline Models**

* Train a basic ML model (e.g., random forest on k-mer frequencies) for a sanity check.
* Then build your first CNN (e.g., 2–3 conv layers + dense regression head).
* Validate using a simple metric like Pearson correlation.

### **🗓️ Days 6–9: Deep Model Optimization**

* Add complexity: deeper CNN, or a hybrid CNN-LSTM if needed.
* Tune hyperparameters (batch size, learning rate).
* Try augmentations like reverse complementing DNA (if needed).

### **🗓️ Days 10–11: Interpretation & Visualization**

* Generate saliency maps, or Grad-CAM if you’re using CNNs.
* Visualize motifs the model finds important (maybe compare with known motifs like from JASPAR).

### **🗓️ Days 12–14: Final Polish**

* Clean up code and notebooks.
* Generate plots: predicted vs actual, saliency maps, loss curves.
* Write up a short report or slide deck summarizing the findings.

## **🧰 Tools & Libraries You’ll Likely Use**

| **Task** | **Tools** |
| --- | --- |
| DNA sequence handling | Biopython, pandas, pyfaidx |
| Expression data | pandas, GEOparse, scanpy (optional) |
| ML/DL Framework | PyTorch or TensorFlow/Keras |
| Visualizations | matplotlib, seaborn, plotly, captum (for PyTorch interpretability) |
| High-performance computing | SLURM for job scheduling, Jupyter on HPC, or batch scripts |

## **🌟 Tips to Succeed in 2 Weeks**

* **Keep the dataset small but meaningful** – maybe ~2,000–5,000 genes for the pilot.
* **Use GPU nodes early** – don’t waste time waiting for CPU models to train if you don’t have to.
* **Modular code** – make each step swappable (e.g., easy to try different sequence encodings or model architectures).
* **Start visualizing early** – don’t wait until the last day to debug model outputs.

If you want, I can help you:

* Choose the best data source for your organism.
* Set up a SLURM job script template for training.
* Sketch out a CNN architecture with specific layer sizes.

Just say the word! 😄

Absolutely — I’ve got you covered! 😊 Since you’re coming from a **computer science background** and stepping into **bioinformatics**, here's a compact, beginner-friendly guide to **gene sequencing** and the relevant **biological concepts** you’ll need for your project.

Think of this as your **"Bio for ML in Genomics" cheat sheet** 🧬💡

## **🧬 1. What Is Gene Sequencing?**

**Gene sequencing** is the process of determining the **order of nucleotides** (A, T, C, G) in a segment of DNA. These nucleotides encode the instructions for building proteins, regulating gene activity, and more.

In modern genomics, we often deal with **sequenced genomes** — the full DNA content of an organism — stored in **FASTA** or **GenBank** format.

## **🧬 2. Key DNA Sequence Concepts You Need**

| **Term** | **Meaning** |
| --- | --- |
| **Gene** | A region of DNA that codes for a functional product (usually a protein). |
| **Promoter** | A sequence upstream of a gene (~−1000 bp) that regulates whether and how much a gene is expressed. |
| **TSS (Transcription Start Site)** | The point at which transcription of a gene begins — the start of the RNA. |
| **Motif** | Short DNA patterns (~6–12 bp) often found in promoters; recognized by transcription factors. |
| **Transcription Factor** | A protein that binds DNA and influences gene expression — sort of like an "on/off" switch. |
| **Expression Level** | How much mRNA is produced from a gene. Measured in **TPM** or **RPKM**. This is your ML target! |
| **Non-coding DNA** | DNA that doesn't make proteins but has regulatory roles (like promoters, enhancers, etc.). |

## **🧬 3. The Central Dogma (Biology’s Info Flow)**

DNA → RNA → Protein

↑ ↑

Promoter Expression level

* You’ll be looking at the **DNA promoter sequence** (the input) and trying to predict the **expression level** (the output).

## **🧬 4. What Is Gene Expression?**

When a gene is **expressed**, it means it is **transcribed** into RNA, which can then be translated into a protein. **Expression levels vary**:

* Across tissues (e.g., brain vs liver)
* Under different conditions (e.g., healthy vs diseased)
* Due to regulatory elements (like promoters and enhancers)

This is the core idea of your project:

Can we **predict the expression level** of a gene from just its **DNA promoter sequence**?

## **🧬 5. Where Does Machine Learning Come In?**

| **ML Concept** | **Biological Parallel** |
| --- | --- |
| **Input features** | One-hot encoded DNA sequence (A, T, C, G) of the promoter |
| **Output label** | Gene expression value (numerical, continuous) |
| **Model type** | Regression model (e.g., CNN for sequences) |
| **Loss function** | MSE or MAE |
| **Evaluation** | Pearson correlation, R² between predicted and true expression |

## **🧬 6. Data Formats You’ll Encounter**

| **Format** | **Content** |
| --- | --- |
| **FASTA** | DNA sequence file (e.g., promoters) |
| **GTF/GFF** | Gene annotation — where genes and promoters are located |
| **CSV/TSV** | Gene expression tables |
| **BED** | Genomic intervals (e.g., promoters, exons) |
| **BigWig** | Compressed genomic signal files (optional for advanced stuff) |

## **🧬 7. How Do You Get Promoter Sequences?**

* From the gene annotations (GTF), get the **TSS**.
* Extract the upstream region, e.g., −1000 bp to TSS.
* Use tools like **BEDTools**, **Biopython**, or **pyfaidx** to grab the DNA sequence from a reference genome.

## **🧬 8. Where Do Expression Values Come From?**

* From RNA-seq experiments.
* Datasets like **ENCODE**, **GTEx**, or **GEO** give you processed expression values in **TPM** (Transcripts Per Million) or **RPKM**.

## **🧬 9. Summary – Your Input/Output**

| **You Use** | **As** |
| --- | --- |
| ATCG... DNA string | 🧬 Input (Promoter sequence) |
| 5.32 TPM | 📈 Output (Expression level) |
| CNN/RNN/Transformer | 🧠 Model |
| Saliency maps | 🔍 Interpret biological motifs learned by your model |

## **🧪 Want to Learn More?**

* [Ensembl Genome Browser](https://www.ensembl.org/index.html) – explore genes and download promoter sequences.
* [UCSC Genome Browser](https://genome.ucsc.edu/) – see genome annotations and sequences.
* [GTEx Portal](https://gtexportal.org/home/) – browse gene expression across tissues.

Let me know if you'd like a **visual diagram**, or even a **small annotated dataset** to start playing with. You're entering a seriously cool intersection of bio and AI — welcome aboard! 😄

Absolutely — and that’s a great mindset. Bioinformatics is **as much about understanding the data** as it is about using it. So let’s break everything down clearly, step by step.

# **🧬 What Are We Doing?**

### **Goal:**

You’re building a dataset to **train a machine learning model** that predicts **gene expression levels** (TPM) using only the **DNA promoter sequence** of each gene.

In other words:  
 *Can I teach a computer to read a gene's "upstream DNA" and guess how much that gene is turned on in the cell?* 🧠🧬

# **🧩 Step-by-Step Breakdown**

## **Step 1: Load Expression Data (TPM values)**

**File:** GTEx\_Analysis\_2017-06-05\_v8\_RNASeQCv1.1.9\_gene\_tpm.gct.gz

### **What's in it:**

* Each **row** = a gene (ID + name)
* Each **column** = a human tissue sample
* Each **cell** = TPM expression value for that gene in that sample

### **What we’ll do:**

* Choose **1 tissue** to focus on (e.g., "Liver")
* Average the TPM values across all liver samples → gives you **one TPM per gene**
* This becomes your **target variable** (like y in ML)

## **Step 2: Parse Gene Annotations (TSS coordinates)**

**File:** gencode.v26.annotation.gtf.gz

### **What's in it:**

* For every gene and transcript: chromosome, start/end position, strand, gene ID
* This lets you find the **TSS (transcription start site)** for each gene

### **What we’ll do:**

* Get the **TSS position** for each gene (for strand +, it's the "start"; for strand −, it's the "end")
* Record:  
  + Chromosome
  + TSS position
  + Strand
  + Ensembl gene ID

This gives you the **coordinates of the promoter** you want to extract from the genome.

## **Step 3: Extract Promoter Sequences**

**File:** hg38.fa.gz

### **What's in it:**

* The actual A/T/C/G DNA sequences for each chromosome

### **What we’ll do:**

* For each gene:  
  + Extract the **1000 bp upstream of the TSS** on the correct strand:  
    - If strand is +: take TSS−1000 to TSS
    - If strand is −: take TSS to TSS+1000, then **reverse complement** the sequence
* Result: you get the **promoter DNA sequence** of the gene

## **Step 4: Join Everything**

### **What we have now:**

* A list of **genes** with:  
  + Promoter sequence (input feature X)
  + TPM expression (target value y)

We merge the promoter sequence table (from Step 3) with the TPM values (from Step 1) using **Ensembl Gene ID**.

## **✅ Final Dataset Format**

| **Gene ID (Ensembl)** | **Promoter Sequence (1000 bp)** | **Expression Value (TPM)** |
| --- | --- | --- |
| ENSG00000123456 | ATCGGTGACCT... | 27.5 |
| ENSG00000234567 | TTAGGCAATGC... | 13.2 |

This is what you’ll feed into your machine learning model to train it.

## **🔍 Why Does This Work?**

Promoters contain **regulatory DNA elements** (called motifs) that control whether a gene is turned on. By looking only at the **sequence** and using **deep learning (e.g., CNNs)**, we can attempt to learn patterns in the sequence that correlate with expression levels.

Let me know if you're ready to:

1. Start with **loading and filtering the expression data**
2. Move on to **extracting TSS positions from the GTF**
3. Or would you like a **Jupyter Notebook template** that combines all of the above?

We’ll go step-by-step, but now you know the whole picture. 😊

Perfect — let’s walk through the full **process step-by-step**, focusing on **what** you need to do, not the code itself. This is your **action plan**, and once you understand this logic, writing or using code will feel much easier.

## **🧠 OVERVIEW: What You're About to Do**

You're going to:

1. Process gene expression data
2. Extract gene positions (TSS) from annotations
3. Extract promoter DNA sequences from the genome
4. Combine everything into one clean dataset

# **🧬 STEP-BY-STEP GUIDE (NO CODE)**

## **🔹 Step 1: Expression Data (TPM values)**

### **What you have:**

The **GTEx TPM file** contains TPM (expression) values for each gene in many tissues and samples.

### **What you do:**

* Unzip and open the file.
* Identify which **samples belong to the tissue** you're interested in (e.g., all liver samples).
* **Average the TPM values** for each gene across those samples.
* Save the result as a table:  
  + Each row = a gene
  + Columns = gene ID, average TPM (for one tissue)

📌 **Now you have your ML target (y)**

## **🔹 Step 2: Gene Annotation (TSS coordinates)**

### **What you have:**

The **GENCODE GTF file** contains start/end positions of all genes, their strand (+/−), and their ID.

### **What you do:**

* Go through the file and extract:  
  + Chromosome
  + Transcription Start Site (TSS):  
    - For + strand: use the **start** coordinate
    - For − strand: use the **end** coordinate
  + Strand
  + Gene ID
* Filter for **protein-coding genes** to keep it relevant.
* Store this in a table: one row per gene with chromosome, TSS, strand, and gene ID.

📌 **Now you know where each promoter is located**

## **🔹 Step 3: Reference Genome (Extract promoter sequences)**

### **What you have:**

The **hg38 FASTA file**, which contains the DNA sequence of each chromosome.

### **What you do:**

* For each gene from Step 2:  
  + Locate the **TSS coordinate** on the correct chromosome.
  + Extract the **1000 base pairs upstream** of the TSS:  
    - If + strand: take 1000 bp *before* the TSS
    - If − strand: take 1000 bp *after* the TSS and then **reverse complement** the sequence
* Make sure to handle edge cases (e.g., when the TSS is too close to the start of the chromosome).
* Save the results:  
  + Each row = gene ID + promoter DNA sequence (string of A/T/C/G)

📌 **Now you have your input feature (X)**

## **🔹 Step 4: Merge Everything**

### **What you do:**

* Join the promoter sequence table (Step 3) with the TPM table (Step 1) using the **gene ID** as the key.
* Filter to keep only genes that appear in both tables.
* Final dataset format:  
  + Gene ID, Promoter Sequence (1000 bp), TPM Value

## **🎯 Your Final Goal**

Create a dataset like this 👇

| **Gene ID** | **Promoter Sequence (A/T/C/G)** | **TPM (Expression)** |
| --- | --- | --- |
| ENSG... | ACTGTTACGGA... | 15.3 |
| ENSG... | TTACCGGTAA... | 42.7 |

📦 Save this as a .csv or .tsv and you’re ready to feed it into your machine learning pipeline.

## **⚠️ Important Reminders**

* Make sure all data is from **the same genome version** (yours is GRCh38 — ✅).
* Use **Ensembl gene IDs** consistently to merge things correctly.
* Account for **strand direction** when extracting promoters (don't skip this).
* Clean up your dataset (e.g., remove genes with missing sequences or zero expression).

Would you like to start with a particular step and have me walk you through how to *implement* it? For example:

* Want to begin with TPM averaging?
* Or start parsing the GTF?

You tell me the starting line, and we go together from there 😄