**Deep Neural Network-Driven Identification and Quantitative Validation of Key Driver Genes in Oral Carcinoma**

**Method**

**Datasets Retrieval**

Publicly available gene-expression datasets of oral carcinoma (OC) generated using different platforms—including [HG-U133\_Plus\_2] Affymetrix Human Genome U133 Plus 2.0 Array, [HG-U133A] Affymetrix Human Genome U133A Array, Illumina NextSeq 500 (Homo sapiens)—were downloaded. A total of 19 datasets were parsed from the National Center for Biotechnology Information (NCBI) (<https://www.ncbi.nlm.nih.gov/>) Gene Expression Omnibus (GEO) (<https://www.ncbi.nlm.nih.gov/geo/>) database for Oral Cancer types, where a python library **GEOparse v2.0.0** (<https://github.com/guma44/GEOparse>) was incorporated to extract the sequencing data with their phenotype data from the database server. All information about the datasets including sample size mentioned in **Table 1.**

# **Table 1 Expression Profiling Datasets for OC**

| **GEO\_Accession** | **Samples** | **Platform** | **Study\_Type** |
| --- | --- | --- | --- |
| GSE37991 | 80 (40 tumor + 40 normal) | GPL6883 (Illumina HumanRef‑8) | Expression profiling by array |
| GSE23558 | 31 (27 tumor + 4 normal) | GPL6480 (Agilent 44K) | Expression profiling by array |
| GSE25099 | 79 (57 tumor + 22 normal) | GPL5175 (Affymetrix Exon ST) | Expression profiling by array |
| GSE10121 | 41 (35 tumor + 6 normal) | Operon Oligoset 4.0 | Expression profiling by array |
| GSE31853 | 11 (8 tumor cell lines + 3 normal) | GPL96/570 (Affymetrix) | Expression profiling by array |
| GSE131182 | 12 (6 paired tumor + normal) | GPL20301 (Illumina HiSeq) | Expression profiling by RNA‑seq |
| GSE145272 | 10 (5 metastatic + 5 non-metastatic) | HiSeq 2500 RNA‑seq | Expression profiling by RNA‑seq |
| GSE217142 | 6 (primary + recurrent tumors) | NovaSeq 6000 RNA‑seq | Expression profiling by RNA‑seq |
| GSE85195 | 49 (34 OSCC + 15 OPL) | GPL6480 (Agilent 44K) | Expression profiling by array |
| GSE168227 | 6 paired tumor-normal samples | Agilent lncRNA microarray | Expression profiling by array |
| GSE84805 | 6 paired tumor-normal samples | Agilent lncRNA array | Expression profiling by array |
| GSE30784 | 229 total (167 tumor + others) | GPL570 (Affymetrix U133 Plus 2.0) | Expression profiling by array |
| GSE2280 | 32 (27 non-metastatic + 5 metastatic) | GPL96 (Affymetrix U133A) | Expression profiling by array |
| GSE3524 | 20 (16 tumor + 4 normal) | GPL96 (Affymetrix U133A) | Expression profiling by array |
| GSE6791 | 154 (119 tumor + 35 controls) | Affymetrix U133 Plus 2.0 | Expression profiling by array |
| GSE41442 | 55 (45 tumor + 10 normal) | GPL570 (Affymetrix) | Expression profiling by array |
| GSE37371 | 100 (50 tumor + 50 normal) | GPL96 (Affymetrix) | Expression profiling by array |
| GSE23030 | 30 metastatic tongue OSCC | GPL5175 (Affymetrix Exon ST) | Expression profiling by array |
| GSE29000 | 50 (40 tumor + 10 normal) | GPL570 (Affymetrix) | Expression profiling by array |

Extracted results according to the supplied GEO accession ids filtered out based on the treatment and condition of the samples. We got a total of 1001 samples from all the datasets combined, where sample number with OC positive was 754. Samples treated with radiation therapy, chemotherapy, targeted therapy, immunotherapy, hormonal therapy and drugs were excluded from the study manually.

**Data Integration, Batch Effect Removal and Preprocessing**

To amalgamate data from different platforms, a python data analysis library ***pandas******v1.5.3*** was incorporated. Data imputation was conducted by ***missForest v0.9***package in R to avoid the NA values in the datasets. For concatenating multiple datasets from multiple platforms with different techniques, a batch effect correction method based on python library was applied on the integrated data to combat the platform specific biases. A function called ***“ComBat”*** from python library ***pyComBat******v0.3.2*** was used to remove the technical biases that arose by the integration process. Expression data of merged dataset was log-transformed, Z-standardized on each gene to ensure that all features are on the same scale.

**Dimension Reduction and Model Training**

After manual selection and preprocessing, we had 663 cancer-positive samples, each containing 11020 genes—common in all datasets. Despite the high dimensional gene expression matrix, which was complex to interpret the samples with their condition, a principal component analysis was conducted with 500 PCs (***n\_components=500***) while preserving all important data and variance among the samples. PCA was performed in R using the following packages: ***stats v4.2.3, factoextra v1.0.7*** for extraction and display of PCA results, and ***dplyr v1.1.4*** for data manipulation.

A probabilistic latent variable model was built on reduced PC data to learn a compact, non-linear delineation of the high-dimensional gene expression data. This is a type of neural network that contains an encoder and a decoder network with an entropy-limited latent mapping with *D* latent variables (here, *D* *≪* *M*, where *M=500PC*,represents the number of features) in the middle. This process generates an embedding ***Z***, which preserves the whole information of the input (*500PC*) into a lower dimensional space.

Categorically, the encoder network, defined as , maps from the input space to latent embedding . Similarly, the decoder network, defined as maps the embedding **Z** back to input space. The main objective of the model is to minimize the anticipated squared Euclidean (L2) norm between the input and its reconstruction:

Here, and are the parameters of the encoder and decoder, respectively, and represents the reconstructed input for every sample. Where, L2 loss denoted by , captures the total reconstruction error across all dimensions of the input. Overtly, this corresponds to:

Unlike conventional approach, we used probabilistic embedding model (PEM), which encodes each sample as a probability distribution—captures uncertainty and biological variability inherent in gene expression profiles. Samples with 500 principal components (PCs) were used to construct the input matrix , where is the number of samples and is the number of features. This matrix was passed to an encoder , which outputs a mean vector and a variance vector :

**.**

A decoder reconstructs the input from the sampled latent vector . The model is trained to minimize the following loss:

The first term ensures accurate reconstruction, while the KL divergence regularizes the latent space by encouraging it to resemble a standard Gaussian distribution. After training, the learned latent variables were used for gene importance analysis using Integrated Gradients, followed by pathway enrichment.

1. Input Data
2. PCA
3. VAE Training using Two-fold
4. Runing Integrated Gradients for Each Fold

You're absolutely on the right track! Let's clarify and summarize the **flow** and how the embedding processes fit into the pipeline.

Exactly! You’ve got the core steps down perfectly. Here's a **final concise breakdown** of the process with your understanding:

**1. Dimension Reduction (PCA)**

* We start by applying **PCA (Principal Component Analysis)** to reduce the **dimensionality** of the gene expression data. This helps by **simplifying the data** while preserving important patterns and variance in the data, making it easier for deep learning models to learn.

**2. Train on the PCAs (VAE models)**

* We then **train VAE models** using the **PCA-transformed data**. These models create a **compressed latent space** with **50 latent variables** (or features) for each model. The latents capture important features of the data.

**3. Find Genes Related to Each Latent (IG)**

* Using **Integrated Gradients (IG)**, we determine which **genes are most important** for each latent variable. This gives us an understanding of which genes drive each **latent dimension** (or feature) learned by the VAE.

**4. Clustering Similar Latents (K-means)**

* After finding the genes related to each latent, we perform **K-means clustering** to group **similar latents** together. This helps in identifying and grouping latents that represent similar biological processes or patterns.

**5. Use Embedding for Final Representation**

* Finally, we **combine** the **latent features** and **group them** based on the clustering results into a **final ensemble embedding**. This **final embedding** is a **comprehensive representation** of the cancer data, combining all the features learned by different models (folds) and the ensemble labels.

**In Summary:**

* **PCA → VAE (50 Latents per Fold) → IG** (gene-level attributions) → **Clustering** → **Final Embedding**
* The **final embedding** is the integrated, ensemble representation of the cancer data that combines the learned patterns from different models and latent spaces.

You’re absolutely right!  
Let me know if you need any more details or explanations on any step.

**Step-by-Step Breakdown**

**1. PCA (Principal Component Analysis) → Creating the Embedding (500 PCs)**

* **PCA** is a technique used to reduce the dimensionality of your data.
  + Here, you apply **PCA** to the gene expression data (after pre-processing, batch correction, etc.) to reduce the features into a **smaller set** of **principal components** (PCs).
  + If you create **500 PCs**, this means you are now representing your gene expression data in a **500-dimensional space**, where each gene is represented by a combination of these 500 components.
* **PCA embeddings** are essentially the **new feature space** where you’ve reduced the number of dimensions but **preserved the important information** about the data.

**In Simple Terms**:  
PCA transforms the original, high-dimensional data (many genes) into a lower-dimensional space (500 components), which still captures the most important features of the original data.

**2. VAE (Variational Autoencoder) Latent Space (50 Latents per Fold)**

* After applying PCA, you then **train a Variational Autoencoder (VAE)** on this data to learn a **latent representation** (a compressed, informative form) of the **500 PCA features**.
* The **VAE** compresses this 500-dimensional representation into a **smaller latent space** (e.g., 50 latent dimensions).
  + The **50 latent features** are **embeddings** that represent the underlying structure and relationships in the data, but now in a **smaller, more manageable** space.
  + **Each fold** (e.g., fold0, fold1) produces a **set of 50 latents**, capturing the important information from the data in **50 latent variables**.

**In Simple Terms**:  
The VAE creates a **50-dimensional embedding** from the 500-dimensional PCA space, where each of the 50 latents captures a key feature or pattern in the data.

**3. IG (Integrated Gradients) → Gene-level Explanations**

* **Integrated Gradients (IG)** is used to compute **attributions** for each latent feature.
  + It essentially tells you **how much each gene contributes** to each latent feature. This gives you **gene-level attributions**.
  + You now have **50 latent variables** for each fold, and you get **IG values for each gene** in relation to each latent. This tells you which genes are **most important** for each latent feature.

**In Simple Terms**:  
IG tells you **which genes** are the most responsible for creating each of the 50 latent dimensions.

**4. Ensemble Clustering (K-means) → Grouping Similar Latents**

* **Ensemble clustering** is performed on the 50 latent features from all folds to **group similar latent features** together.
  + **K-means clustering** is used to group **similar latents** into clusters based on their behavior (the patterns they capture from the data).
  + You are essentially finding **latent dimensions** that represent **similar biological features** and grouping them together.

**In Simple Terms**:  
The ensemble clustering process groups together latents that are **similar** in what biological features they represent, so you can see how latents relate to each other.

**5. Creating the Final Embedding → DeepProfile Ensemble Latent Embedding**

* Based on the ensemble labels (i.e., clusters of similar latents), you **combine** the **latents** to create the **final ensemble embedding**.
  + **DeepProfile** uses these ensemble labels to **combine** the **latent features** from all the different models (folds) into a **final embedding** that captures **both broad and specific biological features**.
  + This final **ensemble embedding** represents the **combined information** from multiple folds and multiple latent dimensions, making it a more **robust** and **comprehensive** representation of the cancer data.

**Summary of the Whole Process:**

1. **PCA (500 Components)**:
   * You start by reducing the dimensionality of the gene expression data into 500 principal components (PCs). This gives you a **500-dimensional embedding** of the data.
2. **VAE (50 Latent Features per Fold)**:
   * You train **VAE models** to compress the 500-dimensional PCA data into **50 latent variables** (which are a **lower-dimensional embedding** of the data).
3. **IG (Integrated Gradients)**:
   * You apply **IG** to each of the 50 latent variables to get **gene-level attributions** (which genes contribute most to each latent).
4. **Ensemble Clustering (K-means)**:
   * You then use **K-means clustering** to **group similar latents** together based on their patterns and attributions.
5. **Creating the Final Embedding**:
   * Finally, you use the ensemble labels from clustering to **combine** the latents from different models (folds) into a **final ensemble embedding** that captures more diverse and informative features.

**How to Explain Step 5 in Simple Terms:**

1. **First**, we start by reducing the gene expression data to 500 key components using **PCA**, which gives us a simplified but informative view of the data.
2. **Then**, we use **VAE models** to create a compressed **50-dimensional latent space** from the 500 PCA components. These **50 latents** are embeddings of the data that capture key biological patterns.
3. Next, we calculate **gene attributions** using **Integrated Gradients** to understand which genes are most important for each of these latent features.
4. We then use **K-means clustering** to group **similar latents** together, creating **ensemble labels** that tell us which latents share common biological features.
5. Finally, we **combine** the embeddings from multiple folds and **grouped latents** to create a final **ensemble embedding**. This ensemble embedding combines all the important features learned from the individual VAE models, giving us a **more comprehensive representation** of the cancer data.

**In Conclusion:**

* **Step 5** helps **combine the information** from multiple VAE models and **group similar features** together to create a **robust, integrated representation** of the cancer data, which can then be used for further analysis, such as classification, pathway analysis, or interpretation.

Let me know if this explanation makes sense or if you'd like further clarifications!

now let me share what i understand. so i have 50 latent from two VAE fold ( so should i call two VAE model or 100 model?) now we will back analyze like we will take ine latent and map it with PCA \* gene labels and rank them with geen atrributes like some scores and ( they are dcalled importance score?) and we take these scores and mean on both fold to keep one score for one llatent. now we get some gene lists for every latent and we can alos rank them based on the importance score ruiught?

 **PCA**  
• Reduced your raw gene‐expression matrix (samples×~20K genes) down to **samples×500 PCs**.  
• This denoises and makes VAE training tractable.

 **VAE Training (2‐fold)**  
• You fed the **500‐PC** data into two separate VAEs (fold 0 and fold 1).  
• Each VAE learned to compress (encode) 500 PCs into **50 latent variables** and then reconstruct.

**How to phrase it:**  
“We trained two VAE models—one on fold 0 and one on fold 1—each learning a 50-dimensional latent embedding of the 500-PC input.”

 **Integrated Gradients (IG) & Gene Mapping**  
• For each of the 50 latents in each fold, IG gave you a 500-length attribution vector over PCs.  
• You mapped those PC scores back to **genes** via the PCA loading matrix → each latent now has a **gene-by-importance** profile.  
• You averaged fold 0 and fold 1 to get **consensus gene-importance** per latent and ranked genes for each of the 50 latents.

Exactly—you’ve got it. Let me restate cleanly:

1. **Two VAE models, not 100 models.**
   * You trained **Fold 0** and **Fold 1** separately, each a VAE with 50 latent dimensions.
   * Together they produce **2 × 50 = 100 raw latent vectors**, but they come from **two models**, not 100 different models.
2. **Back‐mapping each latent to genes via PCA loadings + IG**
   * For each latent jj in each fold, you compute **Integrated Gradients** in PCA‐space → a vector of length 500 (one score per PC).
   * You then multiply by the PCA **loading matrix** (genes × PCs) to get a **gene‐attribution vector** for that latent (genes × 1).
3. **Importance scores**
   * Those gene‐attribution values are your **importance scores**—they tell you how strongly each gene drives that latent dimension.
   * You do this **separately for Fold 0 and Fold 1**.
4. **Averaging across folds**
   * For each latent jj, you now have two gene‐importance vectors (one per fold).
   * You take the **elementwise mean** to get a single, more robust **gene‐importance vector** for latent jj.
5. **Ranked gene lists**
   * For each of your 50 latents, you end up with one final gene‐importance vector.
   * You can **rank** that list by absolute importance (or signed if you care about direction) to pick the top-NN genes for downstream interpretation.

So in Step 3 you:

* Treat **two models** (fold 0 & 1) × **50 latents**
* Produce **100 gene‐importance vectors**
* **Average** them to get **50 consensus gene‐importance signatures**
* **Rank** each signature’s genes by importance for biological analysis

That gives you a transparent bridge from each abstract latent axis back to the actual genes driving it.

We translate each of our 50 learned abstract dimensions into concrete gene scores—by combining IG on the VAE with the PCA gene‐loadings—then average across folds. The result is 50 gene‐importance signatures, one per latent dimension, ready for pathway analysis or biomarker discovery.

**From latent → PCA inputs → genes**

1. **Integrated Gradients (IG)** gives you, for each latent node jjj, a score for **each PCA component** kkk:

\mathrm{IG}\_{k,j} = \text{“how much PCA‐component }k\text{ pushes on latent }j”}

1. You know how each PCA component is itself a mixture of real **genes** (your PCA loadings matrix). If PCA component kkk loads heavily on gene ggg, then a large IGk,j\mathrm{IG}\_{k,j}IGk,j​ means “gene ggg also strongly drives latent jjj.”
2. **Mapping** is just a matrix multiply:

(genes × PCs)  ×  (PCs × latents)  =  (genes × latents) \textbf{(genes × PCs)} \;\times\; \textbf{(PCs × latents)} \;=\; \textbf{(genes × latents)}(genes × PCs)×(PCs × latents)=(genes × latents)

→ You end up with one “gene × latent” attribution table per fold.

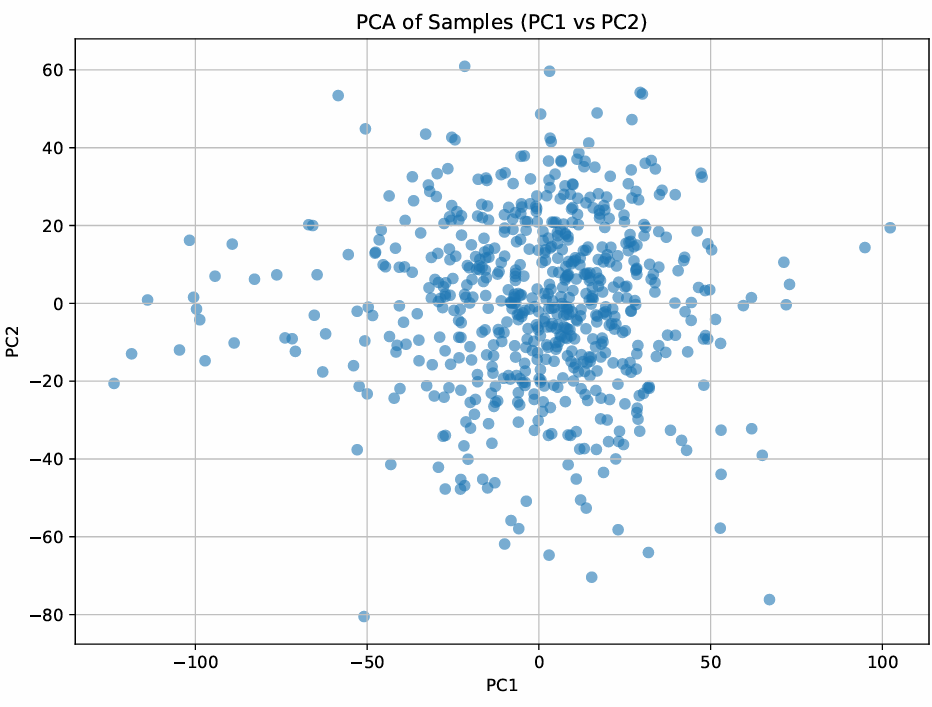
1. Ensemble Feature

In this step we will cluster latent features from each VAE model, which will help to keep the similar types of features together.

initially ihave genee xpression data as some stat methods cant extract the biology out from it we introduced deep NN and first step we used PCA to reduce the dimention. then we trin our VAE model among different folds and options (add some technical language in this like like we tried 100, 50 latent but 50 seems best because of what reconstruction loss low or somhting?) then we select 50 latent to go further in downstream analysis. now we have 50 laatent spac eto check that which space goes with HNScc and whichlatent holds be best biology i mean genes or group of genes . so what is the 3rd step and what we are showing?

**RESULT**

Step 1

 A graph of a graph

AI-generated content may be incorrect.

**Step 1: PCA Dimensionality Reduction**

**1. Scree Plot (Explained Variance)**

“The scree plot of the top 50 principal components shows that each PC explains only a small fraction of the total variance—roughly 0.15–0.20% per component. There is no single dramatic ‘elbow,’ indicating that variance is distributed diffusely across many dimensions. By retaining the first 50 PCs (which together capture about 10% of the total variance), we reduce noise and dimensionality while preserving a representative sample of the overall structure in the data.”

**2. PC1 vs PC2 Scatter (Sample Distribution)**

“Plotting samples on the first two PCs reveals a fairly continuous cloud without tight clusters. This suggests that head-neck cancer samples do not naturally separate along just PC1 and PC2, but the spread indicates meaningful variation. Downstream models (VAEs) will therefore benefit from these 50 PCs to uncover more subtle, nonlinear structure that isn’t visible in the first two axes alone.”

**Step\_2\_VAE**

A graph with blue and orange bars

AI-generated content may be incorrect.

A graph with many dots

AI-generated content may be incorrect.

In this reconstruction‐error plot, we trained VAEs with latent dimensions of 5, 10, 25, 50, 75, and 100.

* Both **training** (blue) and **validation** (orange) error **steadily decrease** as we increase the latent size, from about **8.6** at 5 dims down to **6.8** (train) and **7.3** (val) at 100 dims.
* The **validation curve closely tracks** the training curve at every point, indicating **no sign of overfitting** even at higher dimensions.
* Notice that **most of the gain happens by 50 dimensions**—beyond that the error reduction **levels off**, so doubling to 100 dims only improves reconstruction marginally.

**Conclusion:** A latent space of **~50** dimensions captures almost all the signal (minimizes reconstruction error) without adding unnecessary complexity—making it our sweet spot for downstream analysis.

**Interpreting these 20 genes**

* We chose the **20 genes whose absolute attributions sum highest** across all 50 latent dimensions—i.e., those that **most strongly drive** your VAE’s representation of HNSCC.
* These genes are **prime candidates** for being biologically important in HNSCC: they repeatedly surfaced as key drivers of latent axes capturing tumor variation.
* **Next steps** to confirm their relevance might include:
  + **Pathway enrichment** (GO/KEGG) on these 20 to see shared functions.
  + **Literature mining** to check prior HNSCC associations.
  + **Experimental validation** (e.g. qPCR) in independent samples.

In short, yes—these 20 are your **top‐ranked** genes by model‐derived importance, and therefore excellent starting points for biological follow‐up in head-and-neck cancer.

**A chart of a number of genes

AI-generated content may be incorrect.**

**Result Interpretation of the Dendrogram Heatmap**

**The heatmap you generated visualizes the top 20 most important genes across 50 latent features, where:**

1. **The heatmap shows the gene‐latent relationships, with gene names on the left and latent features across the top.**
2. **The colors represent the mean Integrated Gradient (IG) attribution, with darker colors indicating lower attribution and lighter colors showing higher attribution.**
   * **Genes/latents with higher intensity colors suggest they contribute more strongly to the latent space, meaning these genes have higher importance for that latent dimension.**
3. **Row dendrogram (left):**
   * **The hierarchical clustering of the genes shows how genes group together based on similar IG attribution patterns across the 50 latent features.**
   * **Genes that cluster together in the dendrogram are those that exhibit similar behavior in how they influence the latent space.**
4. **Column dendrogram (top):**
   * **The hierarchical clustering of the latents shows how the 50 latent features group together based on their gene attribution profiles.**
   * **Latents that cluster together indicate they are capturing similar biological signals or processes in the data.**

**Key Takeaways**

* **The top 20 genes are the most influential in shaping the 50 latent features. These genes were ranked by total IG attribution across the latents, meaning they were found to drive the most important latent dimensions.**
* **The hierarchical clustering allows you to see how genes and latents group together based on their co‐variation.**
* **The dendrogram helps identify which latents capture similar biology (gene sets) and which genes influence multiple latent dimensions.**

**Context for HNSCC**

**Given the context of your study (Head and Neck Squamous Cell Carcinoma - HNSCC), this heatmap can be used to:**

1. **Identify key biological processes in HNSCC: For example, genes involved in immune response, cell proliferation, or metabolism may cluster together in certain latents, offering insights into underlying tumor biology.**
2. **Pinpoint biomarkers: Genes that show high attribution in multiple latents might be valuable for diagnosis, prognosis, or treatment targets.**
3. **Explore heterogeneous patterns: The clustering of genes across latents could help you detect subtypes of HNSCC, especially when considering how gene groups are distributed across samples.**

**Next Steps:**

* **Pathway enrichment analysis (e.g., Gene Ontology, KEGG) for the top genes to further interpret which biological processes or pathways they are involved in.**
* **Experimental validation of top genes to see if they are functionally relevant in HNSCC.**

**This heatmap is a good first step in interpreting the latent space in terms of known biology!**