Exemplar Guided Deep Neural Network for Spatial Transcriptomics Analysis of Gene Expression Prediction

Yan Yang¹  Md Zakir Hossain¹  Eric A Stone¹  Shafin Rahman²
¹ BDSI, Australian National University, Australia  ² ECE, North South University, Bangladesh

{Yan.Yang, zakir.hossain, eric.stone}@anu.edu.au  shafin.rahman@northsouth.edu

Abstract

Spatial transcriptomics (ST) is essential for understanding diseases and developing novel treatments. It measures gene expression of each fine-grained area (i.e., different windows) in the tissue slide with low throughput. This paper proposes an Exemplar Guided Network (EGN) to accurately and efficiently predict gene expression directly from each window of a tissue slide image. We apply exemplar learning to dynamically boost gene expression prediction from nearest/similar exemplars of a given tissue slide image window. Our EGN framework composes of three main components: 1) an extractor to structure a representation space for unsupervised exemplar retrievals; 2) a vision transformer (ViT) backbone to progressively extract representations of the input window; and 3) an Exemplar Bridging (EB) block to adaptively revise the intermediate ViT representations by using the nearest exemplars. Finally, we complete the gene expression prediction task with a simple attention-based prediction block. Experiments on standard benchmark datasets indicate the superiority of our approach when comparing with the past state-of-the-art (SOTA) methods.

1. Introduction

Based on an editorial report of the Natural Methods [23], Spatial Transcriptomics (ST) is the future of studying disease because of its capabilities in measuring gene expression of fine-grained areas (i.e., different windows) of tissue slides. However, ST is in low throughput due to limitations in concurrent analysis for the candidate windows [1]. To accurately predict gene expression from each window of a tissue slide image (Fig. 1), this paper proposes a solution named Exemplar Guided Network (EGN), while allowing efficient and concurrent analysis.

In literature, previous works adopt end-to-end neural networks, namely STNet [13] and NSL [8], to establish a mapping between gene expression and the slide image window. STNet is a transfer learning-based approach that fine-tunes a pretrained DenseNet for this task. On the contrary, NSL maps the color intensity of the slide image window to gene expression by a single convolution operation. Though amenable to high throughput because of using neural networks, their prediction performance is inferior.

We investigate two important limitations of the existing approaches [13, 8]: 1) Local feature aggregation: gene expression prediction can be considered as individually grouping the identified feature of each gene type from the slide image window. The long-range dependency among identified features is needed to reason about complex scenarios [35, 22], as those features are generally non-uniformly distributed across the slide image window (see Sec. 3 for details). STNet (i.e., a pure convolution approach) emphasizes local context during feature aggregation. It fails to bring interaction among features that are apart from each other. By experimenting with extensive SOTA architectures, we show that models with local feature aggregation achieve low performance when comparing to models with long-range dependencies. 2) Vulnerable assumption: identifying gene expression of each window of a tissue slide image is with distinct gene expression. Here is an example, we have a tissue slide image with three windows, and each of the windows corresponds with expression of four different gene types. Our goal is to predict the gene expression of each window.
expression by directly detecting the color intensity of the slide image window is vulnerable. By experimenting on standard benchmark datasets, we show this approach only works in extreme cases (for example, in the STNet dataset [13], tumour areas of slide image windows are usually purple, which benefits predicting tumour-related gene expression). This method shows a negative Pearson correlation coefficient (PCC) when evaluating the model with the least reliable gene expression prediction, i.e., PCC@F in Tab. 1.

In this paper, we propose an EGN framework to address the above limitations. EGN uses ViT [9] as a backbone and incorporates exemplar learning concepts for the gene expression prediction task. To enable unsupervised exemplar retrieval of a given slide image window, we use an extractor for defining a representation space to amount similarity between slide image windows. For brevity, we name the extractor output as a global view, which avoids confusion with the ViT representations. Note that, the representation ability of the global view is evidenced in the experiment section. Then, we have a ViT backbone to progressively construct presentations of the given slide image window under long-range dependency. Meanwhile, we have an EB block to revise intermediate ViT representations by the global views of the global view of the given slide image window and exemplars based on the status of the intermediate ViT representations. Semantically, the former update corresponds with ‘what gene expression is known?’, and the latter corresponds with ‘what is gene expression the model wants to know?’. Finally, we have an attention-based prediction block to aggregate the exemplar-revised ViT representations and the global view of the given slide image window to predict gene expression.

Our contributions are summarised below:

- We propose an EGN framework, a ViT-based exemplar learning approach, to accurately predict gene expression from the slide image window.
- We propose an EB block to revise the intermediate ViT representation by using the nearest exemplars of the given slide image window.
- Experiments on two standard benchmark datasets demonstrate our superiority when comparing with SOTA approaches.

2. Related work

This section first review the study of gene expression prediction. Then, we summarise recent achievements of exemplar learning in both natural language processing and computer vision domains.

Gene Expression Prediction. Measuring gene expression is a basic process in developing novel treatments and monitoring human diseases [2]. To increase process accessibility, deep learning methods have been introduced to this task. Existing methods predict the gene expression either from DNA sequences [2] or slide images [13, 8, 30]. This paper explores the latter approach. Meanwhile, these image-based approaches are divided into two streams. First, Schmauch et al. [30] employs a multi-stage method including pretrained ResNet feature extractions [14] and a K-Means algorithm to model the Bulk RNA-Seq technique [21]. It measures the gene expression across cells within a large predefined area that is up to $10^5 \times 10^5$ pixels in a corresponding slide image [30]. However, this approach is ineffective for studies that require fine-grained gene expression information, such as tumour heterogeneity [11]. Second, on the contrary, He et al. [30, 15] and Dawood et al. [8] design a STNet and an NSL framework to predict the gene expression for each window (i.e., fine-grained area) of a slide image. This corresponds with the ST technique [24]. We model ST to predict gene expression, as this potentially solves the bulk RNA-Seq prediction task simultaneously [13]. For example, aggregation of gene expression predictions for each window across a slide image results in a bulk RNA-Seq prediction.

Exemplar Learning. The K-nearest neighborhood classifier is the most straightforward case of exemplar learning. It classifies the input by considering the class labels of the nearest neighbors. Exemplar learning is a composition of retrieval tasks and learning tasks [3]. It has been widely employed to increase the model capability by bringing in extra knowledge of similar exemplars [27, 7, 10, 5, 33, 17, 12, 4, 20, 32, 28, 25]. Patro et al. [27], Chen et al. [7], Teney et al. [31], and Farazi et al. [10] refine the representation of given input with the closest match of text embedding and/or other visual embeddings to help the Visual Question Answering task. Borgeaud et al. [5], Wu et al. [33], Urvashi et al. [4], and Kelvin et al. [12] combine the exemplar learning with transformers for language generation tasks. Their key idea is to enable a more knowledgeable attention mechanism, by picking exemplars as key and value representation. Similarly, Philippe et al. [4] incorporates the attention mechanism with exemplar learning to bring a real-time visual object tracking framework. Other applications of exemplar learning, including fact checking, fact completion, and dialogue, could be found in [20, 32, 28, 25]. However, most of the above approach does not apply to our task because of the domain shift. This paper investigates an application of exemplar learning in gene expression prediction from slide image windows. As a result, we devise an EB block to adapt exemplar learning into our task.
3. EGN Framework

Problem Formulation. We have a dataset collection of tissue slide images, where each image contains multiple windows, and each window is annotated with gene expression. For brevity, we denote the dataset collection as pairs of a slide image window $X_i$ and gene expression $y_i$, i.e., $(X_i, y_i)^N_{i=1}$, where $N$ is the collection size, $H$ and $W$ are the height and width of $X_i$, and $M$ is the number of gene types. We aim to train a deep neural network model to predict $y_i$ from $X_i$.

From the ST study, there exist two main challenges. 1) Long-range dependency: gene expression-related features are non-uniformly distributed over the slide image window $X_i$. Refer to Fig. 3 of [13] for evidence. The interactions among these features are needed to group expression of the same gene type. 2) Skewed gene expression distribution: the expression of some gene types has a skewed distribution, similar to the imbalance class distribution problem. This skewed distribution (see Fig. 3 for an example) poses challenges in predicting expression of these gene types. In this paper, we attempt to mitigate them by learning from similar exemplars.

Model Overview. With these motivations, we have designed the EGN framework (Fig. 2) containing an extractor network for exemplar retrieval and a prediction network that learns from retrieved exemplars. Networks are trained on two stages. 1) Unsupervised exemplar retrieval (Sec. 3.1): we train an extractor $E(\cdot)$ to construct the global view of the slide image window $X_i$, i.e., $e_i = E(X_i)$ and $e_i \in \mathbb{R}^D$, where $D$ is the representation dimension. This global view $e_i$ is used for retrieving the nearest exemplar of $X_i$ to form a set $K_{X_i} = \{e_j, y_j\}_{j=1}^N$, where $N$ is the collection size. We use a vision transformer (ViT) as our backbone. The proposed EB block is interleaved with the vision transformer blocks. With a projector, we refine $e_i$ and $K_{X_i}$ to $h_i$, $\{r^t\}$, and $\{s^t\}$, where $1 \leq t \leq T$, and $T$ is the number of layers. Finally, we have a prediction block that concatenates the refined global view $h_i$ of $X_i$ and the attention-pooled ViT patch representation, to achieve the gene expression prediction task.

3.1. Unsupervised Exemplar Retrieval

To retrieve the exemplar of a given slide image $X_i$ in an unsupervised manner, we propose to train an extractor (i.e., an encoder) with a decoder for image reconstructions. After
expression-related features, by combining a pretrained encoder/extractor with a distance metric to amount the similarity between each pair of slide image windows for the exemplar retrieval. 

**Method.** The StyleGAN [16] applies a style code (i.e., a low dimension vector) to modulate convolution operations during image generations. As indicated by [34], the style code captures both fine-grained and high-level image attributes of a training dataset in an unsupervised manner. Meanwhile, in the style code, each scalar is highly disentangled, i.e., each of them tends to independently control an image attribute. This attribute independency is desired when amounting similarly between images. Thus, we borrow the style code for our unsupervised exemplar retrieval, where the style code is a global view of the given image input. Conventionally, to obtain the style code of a given slide image window, one needs to train a styleGAN and perform a GAN inversion [34] separately. Instead, we jointly train an encoder/extractor $E(\cdot)$ (which replaces the GAN inversion) and a StyleGAN generator $G(\cdot)$, with an image reconstruction objective. Afterwards, we can directly obtain the style code by a simple forward pass in the extractor $E(\cdot)$.

**Objective.** We employ a least absolute deviation loss $L_1$, a LPIPS loss $L_{\text{LPIPS}}$, and a discriminator loss $L_{\text{F}}$ to optimize the image reconstruction, where $F(\cdot)$ is a discriminator. Note that, the reconstruction indirectly impacts the representation ability of the style code. $L_1$ constrains pixelwise correspondence, while $L_{\text{LPIPS}}$ and $L_{\text{F}}$ promotes a better reconstruction fidelity. We have

\[
L_1 = |X_i - G(E(X_i))|,
L_{\text{LPIPS}} = \|\phi(X_i) - \phi(G(E(X_i)))\|_2,
L_{\text{F}} = u(F(X_i)) + u(-F(G(E(X_i)))),
\]

where $\phi(\cdot)$ is a pretrained LPIPS network [37], and $u(\cdot)$ is a Softplus function [16]. Let $X$ be a distribution of slide image windows. The overall objective is

\[
L_E = \min_{G,E} \max_{F} \mathbb{E}_{X_i \sim X} \left[ L_1 + L_{\text{LPIPS}} + L_{\text{F}} \right].
\]

In a later section, we show $E(\cdot)$ effectively encodes the gene expression-related features, by combining a pretrained $E(\cdot)$ with a linear layer for gene expression prediction.

**Exemplar Retrieval.** After convergence, we use the representation space of the extractor $E(\cdot)$ for the exemplar retrieval. Note that, we call the extractor output, the style code, as a global view. The slide image window $X_i$ is encoded into a global view by $e_i = E(X_i)$. Then, we measure the similarity between each given pair of the slide image, for example, $X_i$ and $X_j$, with the Euclidean distance $L_2$, for constructing the nearest exemplar set $K_{X_i}$. We empirically verify the optimal size of $K_{X_i}$ in Sec. 4.3. To generalize the model performance, we restrict that the candidate image pairs are from different patients. The overall process is presented in Fig. 4. In the experiment section, we compare the proposed exemplar retrieval with alternative retrieval approaches.

**3.2. Exemplar Learning**

Our model $C(\cdot, \cdot, \cdot)$ is composed of a projector, a backbone, an EB block, and a prediction block. It maps a slide image window $X_i$, a corresponding global view $e_i$, and a retrieved nearest exemplar set $K_{X_i}$ to gene expression $y_i$, i.e., $y_i = C(X_i, e_i, K_{X_i})$. Our model brings interactions between $K_{X_i}$ and $e_i$ to progressively revise the intermediate representations of $X_i$ in the ViT backbone.

**Projector.** The global views $e_i$ and $e_j \in K_{X_i}$ summarise a wide range of dataset-dependent attributes. We refine the global view to concentrate on the gene expression of interest by several multi-layer perceptrons (MLPs). Firstly, the global view $e_i$ of the given slide image window $X_i$ is projected by $\text{MLP}_h^0(\cdot)$. Secondly, for $e_j \in K_{X_i}$, as the associated gene expression $y_j \in K_{X_i}$ is available, we empower the refinement of $e_j$ by $y_j$. We concatenate $e_j$ and $y_j$ before feeding to $\text{MLP}_h^0(\cdot)$. Thirdly, with $\text{MLP}_h^0(\cdot)$, $y_j$ is projected to the model dimension. We have

\[
h_i^0 = \text{MLP}_h^0(e_i), \quad r_j^0 = \text{MLP}_h^0([e_j, Y_{ij}]), \quad s_j^0 = \text{MLP}_h^0(y_j),
\]

where the superscript of $h_i^0$, $r_j^0$, and $s_j^0$ denotes that they are initial refined global view, and $[\cdot, \cdot]$ is a concatenation operator. $\text{MLP}_h^0(\cdot)$ and $\text{MLP}_h^0(\cdot)$ are two layer perceptrons with a ReLU activation function. $\text{MLP}_h^0(\cdot)$ shares the same parameters with $\text{MLP}_h^0(\cdot)$ but has an extra linear layer on top of it.
Figure 5: EB block architecture. The inputs are $h_i$, $\{r_j^t | j \in T_i\}$, and $Z^t$. Firstly, we project $s_j$ to $s_j^{t+1}$ and have interactions between $s_j^{t+1}$ and $r_j$. This interaction is assisted by $s_j^{t+1}$ and obtains $h_i^{t+1}$ and $r_j^{t+1}$. Secondly, $h_i^{t+1}$ is projected to the number of patches in the ViT backbone, to scale the magnitude of the ViT patch representation $Z^t$. By their interactions, $h_i^{t+1}$ and $Z^t$ are updated to $h_i^{t+1}$ and $Z^t$.

**Backbone.** We use the ViT as our backbone [9]. It has a patch embedding layer and vision transformer blocks. The patch embedding layer tiles the slide image window $X_i$ into non-overlapping patches before flattening and feeding to a linear layer. The outputs are $Z^t = \{z_l^t | l \in 1, \cdots, L\}$, where $L$ is the total number of patches, and $Z^t$ is a matrix representation of the projected patches. Each vision transformer block composes of a self-attention layer and a feed-forward layer, which performs global interaction among $Z^t$. Assuming there are $T$ layers, and let $t \in \{1, \cdots, T\}$, we denote the $t$th path representation at $t$th layer as $z_i^t$.

**EB Block.** This block is interleaved with the vision transformer blocks, which brings $s_j$, i.e., knowledge about gene expression of the nearest exemplar, to each ViT patch representation $z_i^t$. Firstly, we project $s_j$ (i.e., a projection of gene expression of the exemplars) to $s_j^{t+1}$. With $s_i^{t+1}$, we have interactions between $h_i^t$ and $r_j$ (i.e., refined global views) to obtain $h_i^{t+1}$ and $r_j^{t+1}$. As $h_i^{t+1}$ and $r_j^{t+1}$ are initialized from the same extractor $E(\cdot)$, their interactions could serve as a bridge for the revision of the ViT patch representation. Secondly, considering $h_i^{t+1}$ is initially extracted by the reconstruction-based network $E(\cdot)$, $h_i^{t+1}$ is expected to contain spatial information about the input image distribution. With $h_i^{t+1}$, we revise the ViT patch representation $z_i^t$, obtaining $h_i^{t+1}$ and $z_i^{t+1}$. This revision process brings interactions between $h_i^{t+1}$ and $z_i^{t+1}$ to propagate information about possible gene expression (which receives from the exemplars) and patch representation statuses. The overall process is shown in Fig. 5.

To do so, firstly, we concatenate $h_i^t$ and the difference $h_i^t - r_j^t$ to bring interactions between $h_i^t$ and $r_j^t$, with $MLP_m(\cdot)$. We chunk the outputs to $m_{h,j}$ and $m_{r,j}$. Then, they are fused with $s_j$ to retrieve gene expression from $s_j^{t+1}$ (a projection of $s_j^t$). Semantically, $m_{h,j}$ summarises the existing knowledge of gene expression, and $m_{r,j}$ tells the desired knowledge. Mathematically, we have

$$
\hat{h}_i^{t+1} = h_i^t + \text{Avg}([m_{h,j} \cdot s_j^{t+1} | j \in T_i]),
$$

$$
r_i^{t+1} = r_i^t + m_{r,j} \cdot s_j^t,
$$

$$
m_{h,j}, m_{r,j} = \text{Chunk}(\sigma(MLP_m(h_i^t, h_i^t - r_i^t))),
$$

$$
s_j^{t+1} = MLP_s(s_j^t)
$$

where $MLP_m(\cdot)$ is a Multi-layer perceptron, the Chunk(\cdot) operator equally splits the input into two outputs, and $\sigma(\cdot)$ is a Sigmoid function. To avoid notation overloading, we remain to use $MLP_s(\cdot)$ as a single-layer perceptron. Secondly, $h_i^{t+1}$ and $Z^t$ reciprocate each other by scaling the magnitude of each patch representation $z_i^t \in Z^t$:

$$
Z^{t+1} = Z^t + MLP_x(O_h^t),
$$

$$
h_i^{t+1} = h_i^{t+1} + MLP_x(\text{Avg}(O_h^t)),
$$

$$
O_h^t, O_s^t = \text{Chunk}(MLP_o(Z^t) \cdot \sigma(MLP_x(h_i^{t+1}))),
$$

where $MLP_x(\cdot)$, $MLP_o(\cdot)$, and $MLP_y(\cdot)$ are single-layer perceptrons. $O_h^t$ and $O_s^t$ are the fusion of the ViT patch representations $Z^t$ and $h_i^{t+1}$. Note that, the output dimension of $MLP_o(\cdot)$ is the number of patches in the ViT, and each scalar of the output is used to scale a corresponding patch from $MLP_x(Z^t)$. By scaling the magnitudes of each ViT patch representation, we indirectly inject the knowledge about gene expression of the exemplars, while updating the refined global view $h_i^{t+1}$ to facilitate both interactions with the exemplars and the ViT representation in following layers. One may note that the current operation is single-head based, which can be extended to a multi-head operation by expanding the output dimension of $MLP_o(\cdot)$ in practice.

**Prediction Block.** Following [38], each ViT patch representation shows different priorities toward the gene expression prediction task. We apply an attention pooling layer $AttPool(\cdot)$ to aggregate $z_i^T \in Z^T$. Moreover, $h_i^T$ is a progressively refined global view of the slide image window $X_i$, which captures high-level attributes of $X_i$. We concatenate $h_i^T$ and $AttPool(Z^T)$ for the prediction. We have

$$
y_i = MLP_g([h_i^T, AttPool(Z^T)]),
$$

5043
Table 1: Quantitative gene expression prediction comparisons with SOTA methods in STNet dataset and 10xProteomic dataset. We bold the best results. We use '-' to denote unavailable results. Models are evaluated by four-fold cross-validation and three-fold cross-validation in the above datasets. Our proposed EGN framework consistently outperforms the SOTA methods in MAE_{x10^3}, PCC@F_{x10^3}, PCC@S_{x10^3} and PCC@M_{x10^3} for both datasets. The CycleMLP finds the best MSE_{x10^2} in the 10xProteomic dataset.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE_{x10^2}</td>
<td>MAE_{x10^3}</td>
<td>PCC@F_{x10^3}</td>
<td>PCC@S_{x10^3}</td>
<td>PCC@M_{x10^3}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STNet Dataset</td>
<td>4.52</td>
<td>-</td>
<td>4.28</td>
<td>4.41</td>
<td>4.49</td>
<td>4.53</td>
<td>4.46</td>
<td><strong>4.10</strong></td>
</tr>
<tr>
<td></td>
<td>1.70</td>
<td>-</td>
<td>1.67</td>
<td>1.68</td>
<td>1.70</td>
<td>1.71</td>
<td>1.69</td>
<td><strong>1.61</strong></td>
</tr>
<tr>
<td></td>
<td>0.11</td>
<td>-</td>
<td>0.97</td>
<td>1.11</td>
<td>0.91</td>
<td>0.99</td>
<td>0.87</td>
<td><strong>1.51</strong></td>
</tr>
<tr>
<td></td>
<td>1.82</td>
<td>-</td>
<td>1.86</td>
<td>1.95</td>
<td>1.54</td>
<td>1.74</td>
<td>1.72</td>
<td><strong>2.25</strong></td>
</tr>
<tr>
<td></td>
<td>1.74</td>
<td>-</td>
<td>1.72</td>
<td>1.79</td>
<td>1.74</td>
<td>1.79</td>
<td>1.74</td>
<td><strong>2.02</strong></td>
</tr>
<tr>
<td>10xProteomic Dataset</td>
<td>12.40</td>
<td>-</td>
<td>7.54</td>
<td><strong>4.69</strong></td>
<td>5.45</td>
<td>5.25</td>
<td>5.04</td>
<td>5.49</td>
</tr>
<tr>
<td></td>
<td>2.64</td>
<td>-</td>
<td>2.27</td>
<td><strong>1.55</strong></td>
<td>1.56</td>
<td>1.65</td>
<td>1.66</td>
<td><strong>1.55</strong></td>
</tr>
<tr>
<td></td>
<td>1.25</td>
<td>-</td>
<td>5.11</td>
<td>5.88</td>
<td>6.40</td>
<td>5.46</td>
<td>5.59</td>
<td><strong>6.78</strong></td>
</tr>
<tr>
<td></td>
<td>2.26</td>
<td>1.84</td>
<td>4.64</td>
<td>6.60</td>
<td>7.15</td>
<td>6.35</td>
<td>6.36</td>
<td><strong>7.21</strong></td>
</tr>
<tr>
<td></td>
<td>2.15</td>
<td>0.25</td>
<td>4.90</td>
<td>6.32</td>
<td>6.84</td>
<td>6.04</td>
<td>6.00</td>
<td><strong>7.07</strong></td>
</tr>
</tbody>
</table>

where MLP_g is a single-layer perception.

**Objective.** We optimize C(·, ·, ·) with mean squared loss (i.e., the Euclidean distance) L_2 and batch-wise PCC L_{ppc}. We have

\[ L_{total} = L_2 + L_{ppc} \]

### 4. Experiments

**Datasets.** We perform experiments in the publicly available STNet dataset [13] and 10xProteomic datasets\(^1\). The STNet dataset contains roughly 30,612 pairs of the slide image window and the gene expression. This dataset covers 68 slide images from 23 patients. Following [13], we target predicting expression of 250 gene types that have the largest mean across the dataset. The 10xProteomic dataset has 32,032 slide image windows and gene expression pairs from 5 slide images. We select target gene types in the same way as the STNet dataset. We apply log transformation and min-max normalization to the target gene expression. Note that, our normalization method is different from [13] (they use log transformation and a custom normalization, i.e., dividing the expression of each gene type by the sum of expression from all gene types, for each slide image window). Our normalization method allows independent analysis of gene expression prediction.

**4.1. Experimental set-up**

**Baseline methods.** We compare with extensive SOTA methods from gene expression prediction, ImageNet classification benchmarks, and exemplar learning.

- STNet [13] and NSL [8]. They are the SOTA methods in gene expression prediction.
- ViT [9], MPViT [19] and CycleMLP [6]. We use the SOTA ImageNet classification methods in our task. They are strong baselines in our task. Specifically, we use ViT-B, MPViT-Base, and CycleMLP-B2. Please refer to [9, 19, 6] for details.
- Retro [5] and ViTExp. We explore the SOTA exemplar learning methods. However, Retro is originally developed for natural language processing. We adapt it by providing the extractor output \( e \) as the exemplar representations. ViTExp directly concatenates the exemplar representations to the ViT patch representation. The exemplar representations are added with an embedding to be differentiated from the patch representation of the slide image window. Both Retro and ViTExp are based on the ViT-B architecture [9].

**Evaluation metrics.** We evaluate the proposed methods and alternative baselines with PCC, mean squared error (MSE), and mean absolute error (MAE). We use PCC@F, PCC@S, and PCC@M to denote the first quantile, median, and mean of the PCC. The PCC@F verifies the least performed model predictions. The PCC@S and PCC@M measure the median and mean of correlations for each gene type, given predictions and GTs for all of the slide image windows. Meanwhile, the MSE and MAE measure the sample-wise deviation between predictions and GTs of each slide image window for each gene type. Note that, for MSE and MAE, the lower value means better performance. In contrast, for PCC@F, PCC@S, and PCC@M, the higher value indicates better performance.

**Implementation details\(^2\).** The architectures and optimization settings for our extractor follow [36]. For the exemplar learning, we implement EGN by using the Pytorch frame-

\(^1\)https://www.10xgenomics.com/resources/datasets

\(^2\)Codes are available at: https://github.com/Yan98/EGN
The majority of gene types covered by the first quantile have the first quantile of PCC across all gene types. Note that, the ric evaluates the worst model capability, by calculating the significantly outperforms the baseline methods. This metric evaluates the worst model capability, by calculating the skewness of expression distributions, which are the most challenging part of the prediction task. Our method has 0.038 higher than the second-best method in PCC@F; and 4) STNet and NSL even shows a negative correlation with PCC@F. Again, the gene expression-related feature is usually non-uniformly distributed across the slide image window input. They do not have long-range interactions to capture the expression of the same gene type from these features. Moreover, NSL even shows a negative correlation with PCC@F. This validates our claims that predicting gene expression directly with the color intensity is vulnerable, and it is only feasible in extreme cases (recall the example of tumour-related gene expression in Sec. 1).

4.2. Experimental results

We compare our EGN framework with the baselines on the STNet dataset and the 10xProteomic dataset (Tab. 1). As the gene expression prediction task emphasizes capturing the relativity variation, we bias on the PCC-related evaluation metrics, i.e., PCC@F, PCC@S, and PCC@M. Our EGN consistently achieves the SOTA performance in MAE, PCC@F, PCC@S, and PCC@M. Our findings are as follows: 1) It’s worth noting that Retro and ViTExp use the ViT as the backbone. However, their performance is lower than the ViT. Both Retro and ViTExp use attention-based operations to directly have interactions between the ViT representations and the global views of the exemplars, when comparing with our EB block. This validates the necessity of using the global view of the input slide image window as a bridge to introduce the knowledge of exemplars to the ViT representations; 2) CycleMLP and MPViT, the SOTA methods in the ImageNet classification task, lead to the second-best performance in the STNet dataset and 10xProteomic dataset in PCC-related evaluations. Meanwhile, CycleMLP finds the best MSE in 10xProteomic. Note that, PCC-related evaluation metrics are most important in our task. By using the exemplars, our model that uses the vanilla ViT backbone outperforms them with a reason marginal in PCC-related evaluation metrics, while overall achieving similar MSE and MAE with them; 3) The PCC@F of our model significantly outperforms the baseline methods. This metric evaluates the worst model capability, by calculating the first quantile of PCC across all gene types. Note that, the majority of gene types covered by the first quantile have skewed expression distributions, which are the most challenging part of the prediction task. Our method has 0.038 - 0.040 higher than the second-best method in PCC@F; and 4) STNet and NSL even shows a negative correlation with PCC@F. This validates our claims that predicting gene expression directly with the color intensity is vulnerable, and it is only feasible in extreme cases (recall the example of tumour-related gene expression in Sec. 1).

Quantitative Evaluation. We present the latent space visualization (Fig. 6), by considering the top performed models from Tab. 1 and Tab. 4 (see details of ‘Backbone only’ and ‘w/o Projector’ settings in Sec. 4.3). Note that, the ‘Backbone only’ setting has a smaller number of parameters than a regular ViT-B. The extra labels (i.e., tumour and normal) from the STNet dataset are used for annotations. To enable a clean visualization, we randomly sample 256 representations of the slide image window for each label, i.e., tumour and normal. By gradually using the proposed components (from Fig. 6 (c) to Fig. 6 (e)), i.e., introduction exemplars to our model, our method sufficiently separates the tumour representations from the normal representations, by having an improved gene expression prediction than alternative approaches.

4.3. Ablation study

We study the capability of each model component by conducting a detailed ablation study in the STNet dataset. Extractor Capability. We explore alternative approaches for retrieving exemplars (Tab. 2). We compare with representations from AlexNet [18] and ResNet50 [14]. Note that, they are pretrained on ImageNet, and our E(·) is learned in an unsupervised manner. We explore diverse distance matrices including LPIPS [37], L2, L1, and cosine (i.e., cosine similarity). We have the following findings: 1) LPIPS distance retrieved exemplars lead to bad model performance. This distance is trained based on the human perception of regular images, where these images are different from the
slide image windows. Thus, it retrieves bad exemplars and damages the model performance; 2) Retrieving exemplars with ResNet50 lead to decent performance, though the pretrained ResNet50 lacks gene expression-related knowledge. This retrieval method complements the knowledge about image textures learned from the ImageNet for our task, which enhances the versatility of our model representation and verifies our model robustness; and 3) Using the extractor $E(\cdot)$ with $L_2$ distance achieves the best performance.

**EB Block Settings.** Firstly, we present PCC@M (Fig. 7(a)), MSE (Fig. 7(b)), and MAE (Fig. 7(c)), by varying the number of exemplars used in the EB block from one to fifteen. Having nine exemplars find the best PCC@M and MAE, and we have the best MSE by using ten exemplars. Again, our task emphasizes capturing relative changes. Thus, our final recommendation is to use nine exemplars. Secondly, we ablate the EB block architectures and the frequency of interleaving with the vision transformer blocks (Tab. 3). This balances the model’s capability in structuring the ViT representations and receiving knowledge from the exemplars. We have the best PCC@M, by setting heads, head dimension, and frequency to 8, 64, and 2.

**Model Architectures.** We present the performance of ‘Pretrained $E(\cdot)$’, ‘Backbone only’, ‘w/o EB block’, and ‘w/o projector’ in Tab. 4. ‘Pretrained $E(\cdot)$’ adds one trainable linear layer to $E(\cdot)$, while freezing weights of $E(\cdot)$ for gene expression prediction. This is also known as linear probing in the literature. ‘Backbone only’ uses the ViT backbone architecture. Note that, we use a non-regular ViT architecture which is determined by a binary search. ‘w/o EB block’ is equivalent to concatenating the pooled ViT representations with the global view for predicting gene expression. ‘w/o projector’ replaces the projector with a linear layer to unify the dimension. Our findings are as follows: 1) The ‘Pretrained $E(\cdot)$’ setting has better performance than the STNet and NSL (Tab. 1). Our $E(\cdot)$ captures gene expression-related features in an unsupervised manner, showing the potential of our unsupervised exemplar retrieval; 2) The ‘w/o EB block’ setting achieves worse performance than the ‘Backbone only’ setting. Concatenating the global view for gene expression prediction with a single linear layer disables interactions between the global view and the ViT representations. This potentially causes inconsistent behaviors and results in poor performance; and 3) With all proposed components, we have the best performance.

### 5. Conclusion

This paper proposes an EGN framework to accurately predict gene expression from each fine-grained area of tissue slide image, i.e., different windows. EGN uses the ViT as a backbone while integrating with exemplar learning concepts. We first have an extractor to retrieve the exemplars of the given tissue slide image window in an unsupervised manner. Then, we propose an EB block to progressively revise the ViT representation by reciprocating with the nearest exemplars. With extensive experiments, we demonstrate the superiority of the EGN framework over the SOTA methods. EGN is promising to facilitate studies on diseases and novel treatments with accurate gene expression prediction.
References


