Multi-Task Compound Activity Modeling with High Content Imaging

Multi-Task Compound Activity Modeling with High Content Imaging

Master Thesis

Hsiu-Chi Cheng

October 14, 2024

Advisors: Dr. Pushpak Pati, Johnson & Johnson Principal Data Scientist
Prof. Dr. Julia Vogt, ETH Zurich Assistant Professor

Department of Computer Science, ETH Zürich

Abstract

High Content Imaging (HCI) has transformed drug discovery by enabling high-throughput screening and providing detailed phenotypic data on the cellular effects of compounds. However, analyzing the complex, high-dimensional HCI data remains a challenge. Deep Learning (DL) has emerged as a powerful tool to address this, offering automated feature extraction that improves analytical accuracy. Despite these advances, most DL applications in drug discovery focus on single-task models, where each task, such as mechanism of action (MOA) prediction or toxicity assessment, is modeled independently. This approach is inefficient and overlooks the potential for shared information across tasks.

In this thesis, we present a multi-task learning (MTL) framework that jointly trains models for multiple related tasks, improving efficiency and performance by leveraging shared representations. Our approach also incorporates attention-based Multiple Instance Learning (MIL), which enhances interpretability and performance by focusing on the most relevant cellular instances within each HCI image.

We apply this method to key drug discovery tasks, supported by auxiliary tasks, to improve model generalization and robustness. Through careful task weighting and optimization, we demonstrate the potential of MTL to streamline modeling in drug discovery, offering a scalable solution that can handle the complexity of HCI data.

Contents

Chapter 1

Motivation

Drug Discovery and the Role of HCI In the field of drug discovery, understanding the diverse properties of chemical compounds is essential. The traditional drug discovery process is often extensive, involving time-consuming and labor-intensive experiments that can be both costly and inefficient. HCI has streamlined this workflow by enabling the automated, high-throughput screening of compounds, providing detailed phenotypic data at the cellular level. HCI facilitates tasks such as mechanism of action (MOA) prediction, toxicity assessment, and morphological profiling, making it an indispensable tool for image-based phenotypic drug discovery [4]. The rich, high-dimensional nature of HCI data, however, poses significant challenges for analysis, which has led to the increasing adoption of DL techniques.

The Impact of Deep Learning in Drug Discovery Recent advancements in deep learning have significantly impacted various stages of drug discovery, from early-stage hit identification to optimization and safety assessment [2][14][16][3]. DL models, particularly convolutional neural networks (CNNs), are capable of automatically extracting complex features from high-dimensional HCI data, thereby enhancing analytical accuracy and generalization compared to traditional methods [1].

Challenges of Single-Task Learning in Drug Discovery However, most DL applications in drug discovery have focused on single-task learning. In these cases, each task is handled independently, requiring a separate model for each task. While this approach can be effective, it is computationally intensive and does not scale well when multiple properties need to be evaluated simultaneously. Furthermore, the scarcity of labeled data for individual tasks limits model performance, making it challenging to train robust models. This paradigm also overlooks the potential relationships between tasks, such as

correlations in cellular responses across different phenotypic assays, which, if leveraged, could lead to more generalized models and reduce the need for task-specific datasets.

Multi-Task Learning Multi-task learning (MTL) presents a compelling alternative by allowing a single model to learn multiple tasks concurrently, using shared representations to capture the underlying relationships between them. This is particularly useful in drug discovery, where many tasks may share common features or biological mechanisms [19]. By training on multiple related tasks simultaneously, MTL can improve performance across tasks, even when individual datasets are limited. Techniques such as task-specific loss weighting, gradient-based optimization, and mixture of experts models have been developed to address the challenges of MTL, ensuring balanced learning across tasks without negative interference [15][17][18].

Attention-Based Multiple Instance Learning (MIL) in HCI The complexity of HCI data, where a single image may contain multiple instances of cellular responses, makes it a natural candidate for MIL. MIL frameworks can effectively handle the ambiguity in labeling by focusing on instances that contribute most to the overall assessment. Incorporating attention mechanisms into MIL models further enhances their capacity to identify the most relevant instances within a high-dimensional dataset, improving both performance and interpretability [6][12][11]. For example, MixMIL has recently demonstrated state-of-the-art performance on HCI data by using an attention-based MIL approach to accurately classify drug mechanisms of action [5].

Our Approach: Multi-Task Learning with HCI for Drug Discovery In this study, we aim to leverage multi-task learning in conjunction with HCI data to address the challenges of traditional single-task models in drug discovery. We focus on six primary tasks while utilizing auxiliary tasks to enhance the model's generalization and robustness. Our approach integrates attention-based MIL frameworks to effectively handle the complex and ambiguous nature of HCI data, while multi-task learning allows us to exploit shared information across tasks, improving overall performance and efficiency. Chapter 2 of this thesis provides a detailed description of the dataset and experimental setup. Chapter 3 delves into our MTL methodology, including the task-weighting strategy and network architecture. Chapter 4 presents the experimental results, and Chapter 5 concludes with a discussion of our findings and their implications for future research in drug discovery.

Chapter 2

Dataset

2.1 Data Introduction

To characterize compound properties in the early stages of drug discovery, compounds are applied to cells at specific concentrations and observed over designated time points, followed by various assays. These compounds are typically distributed across wells in a plate, alongside control wells and different compound samples. To ensure accuracy and reproducibility, the same compounds may be applied across multiple plates. Each well can be further analyzed by capturing several Field of View (FOV)s. Based on these observations, the properties of the compounds can be assessed. A representative sample of this process is illustrated in Figure 2.1.

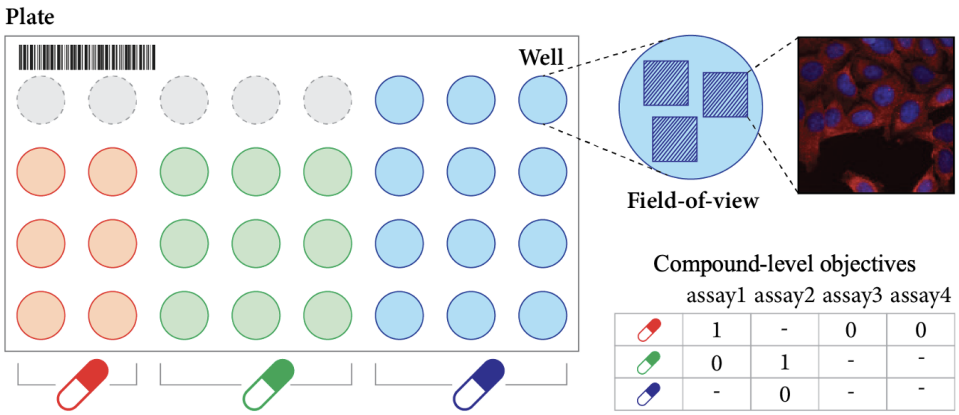

Figure 2.1: Experiments and labels.

	Wells	Compounds	Neg. wells	Pos. wells	Neg. comp.	Pos. comp.
T1	1,125	472	418	707	192	280
T2	1,386	659	815	571	398	261
T3	2,271	930	312	1,959	151	779
T4	1,136	692	535	601	398	294
T5	1,492	422	796	696	222	200
T6	7,130	3,244	3,668	3,462	1,666	1,578

Table 2.1: Numbers of task-specific wells and compounds.

2.2 Data Statistics

In this study, we utilize a internal private dataset, which is central to our investigation of drug discovery processes. The dataset comprises 336,075 FOV high-content images captured at a fixed time point following the application of compounds to the U2OS cell line. Additionally, the dataset includes partially annotated labels for 200 distinct tasks at the compound level with each label being either True or False.

These images are derived from experiments involving 41,843 compounds distributed across 84,030 wells, with only 4% of the compound-task pairs being labeled. Given the challenge of optimizing a large and sparsely annotated dataset, we collaborated with the biology team to select the six most critical tasks for focused analysis. These tasks are referred to as Task-1 (T1), Task-2 (T2), Task-3 (T3), Task-4 (T4), Task-5 (T5), and Task-6 (T6) in this study.

The total number of annotations for these six tasks is presented in Table 2.1, where wells or compounds may be repeated across different tasks. Despite the overall size of the dataset, the most important tasks contain only a few hundred data points each. Considering the difficulty of training deep learning models with limited data, we decided to train models at the well level and make final assessments at the compound level. Specifically, during the training and validation processes, we assign the same compound-task annotation to all well-task labels if the wells originate from the same compound. During inference on the testing data, we aggregate all wells associated with the same compound and utilize a voting mechanism to determine the final result.

2.3 Cross-Validation Strategy

To ensure the stability of our experiments, we divided the dataset into 5 folds, ensuring separation at the compound level while adhering to biological logic. All AUC results presented in this paper are the mean values calculated across these 5 folds.

Chapter 3

Methods

In this section, we detail the deep learning pipelines employed in our study. We begin by outlining the process used to extract FOV-level features, which serve as the foundation for our analysis. Following this, we explore two distinct strategies designed to address the challenges of multi-task classification, providing a comprehensive overview of their implementation and evaluation.

3.1 Feature extraction

Our feature extraction workflow, depicted in Figure 3.1, consists of three main modules: preprocessing, foundation model, and postprocessing. The input FOV image, with a resolution of 960 × 960, is initially divided into smaller crops, with the number of crops treated as a hyperparameter, tested among the values [4, 9, 16, 25]. The cropping strategy is illustrated in Figure 3.2. Each crop undergoes image preprocessing steps, including data

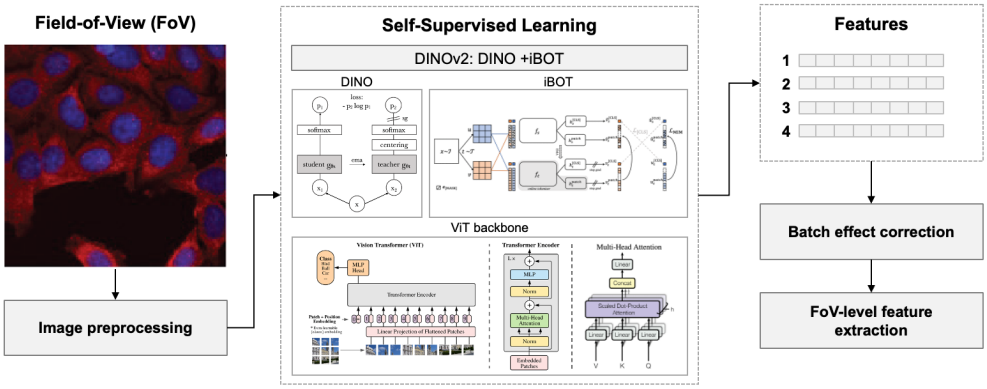

Figure 3.1: Feature extraction workflow.

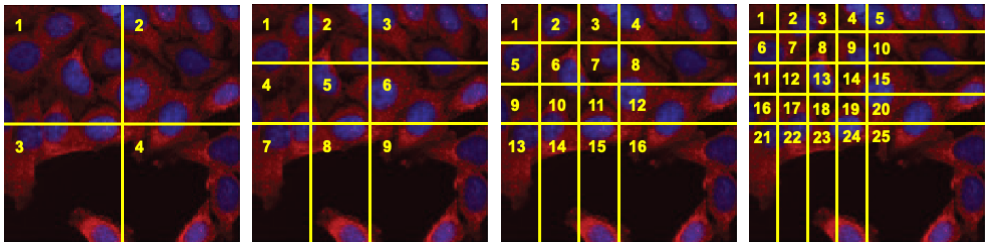

Figure 3.2: Example stride size

augmentation and normalization. These processed crops are then input into the self-supervised learning model DINOv2 [13], producing corresponding feature embeddings, each with a dimension of 768. Given that the images are sourced from multiple experiments, addressing batch effects is essential [10] [9]. To mitigate this, our pipeline incorporates Median Absolute Deviations (MAD) batch correction at the plate level, aligning low control intensity values across all plates and adjusting the corresponding sample values accordingly.

3.2 STL Classification

In this section, we address the approach taken to solve the single-task multi-instance learning problem. For each task, a comprehensive model was constructed to make classifications, encompassing modules for compound representation, pre-attention, attention, and classification, as illustrated in Figure 3.3. The subsequent paragraphs will provide a detailed explanation of each module.

Compound representation As discussed in Chapter 2, this study focuses on utilizing well-level features during the training process, rather than compound-level features. To achieve this, all FOVs associated with a well are aggregated, and the corresponding crop features for these FOVs are compiled into an array representing the well. The embedding sizes of different wells may vary, as the number of FOV is not uniform across all wells. For example, if a well w has four FOVs (f^1, f^2, f^3, f^4) and four crops for each FOV^i $(c_1^i, c_2^i, c_3^i, c_4^i)$, then w can be represented by $(c_1^1, c_2^1, ..., c_4^4)_{16x768}$.

Pre-attention The pre-attention module is designed to enhance feature extraction and map the pre-trained embeddings to a suitable dimensionality for downstream task learning. Given the limited size of our dataset, employing a highly complex architecture is not warranted. Therefore, we implement a single fully connected layer, followed by an activation and a Dropout layer. In our experiments, we tested various output dimensions, including 256, 128, 64, 32, and 16

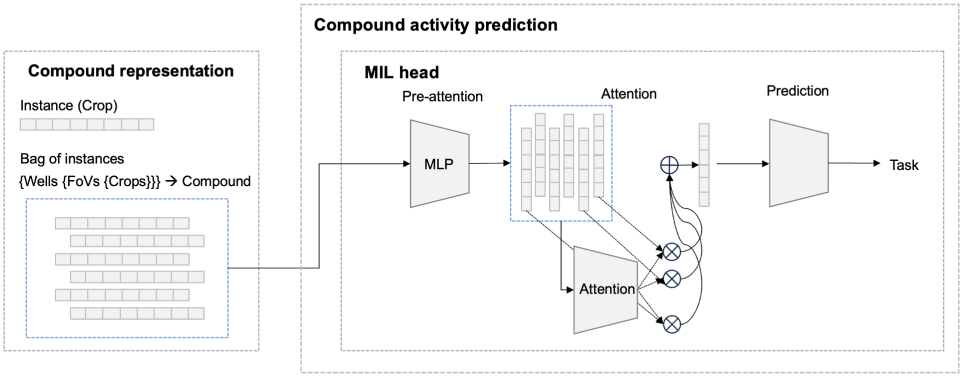

Figure 3.3: STL model structure.

Attention The attention mechanism aids in identifying the most relevant features for making classification decisions. In this section, we aim to conduct experiments in a systematic and exploratory manner to gain a comprehensive un

Model selection Understanding of the models that best suit our dataset. This will provide a foundation for selecting models and hyperparameters in future multi-task experiments. Specifically, we experiment with ABMIL [8] and DSMIL [11] with gated attention [8] mechanisms. Additionally, we explore different model sizes ranging from 20,000 to 1,000,000 trainable parameters, various learning rates from 1e-3 to 1e-7, different numbers of layers, and a variety of activation functions, including ReLU and GELU [7].

Classification The design of the classification component varies depending on the model. For ABMIL, it consists of fully connected layers followed by activation functions and dropout layers, while in the case of DSMIL, it involves a one-dimensional convolutional layer. In both instances, the output is a logit, which is subsequently transformed into a probability using the softmax function. This probability indicates whether the task is present (1) or not (0).

Early stopping By default, each model is trained for 100 epochs. However, in some cases, the model stops updating after just a few epochs, which can unnecessarily occupy GPU resources. To optimize resource usage, we introduce two parameters: the number of warm-up epochs and the patience value. The model must run for at least the specified number of warm-up epochs. After that, if the validation AUC does not improve for a consecutive number of epochs defined by the patience value, the training is terminated early to free up resources.

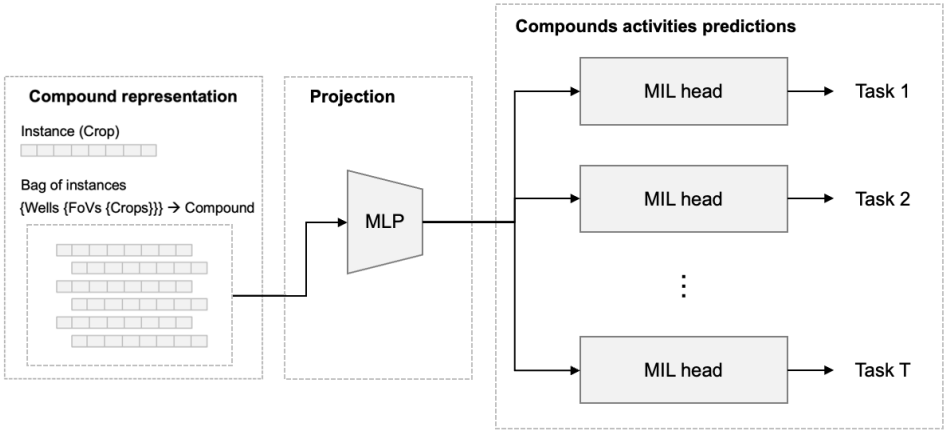

Figure 3.4: MTL model structure.

3.3 MTL Classification

After conducting targeted experiments on single-task problems, we narrowed the hyperparameter search space. Specifically, we focus on the DSMIL model, exploring learning rates between 1e-4 and 1e-6, testing different crop numbers [4, 9, 16], and experimenting with three distinct model sizes. In the following sections, we provide a detailed overview of the core model architectures, alongside the various experiments and design choices. This includes our strategy for training on large datasets, the selection of task groups, and modifications to the training loss functions.

3.3.1 Model structure

As illustrated in Figure 3.4, the multi-task models extend the single-task model by incorporating a shared projection backbone and individual heads for each task. The projection backbone is designed to utilize the full dataset, potentially enhancing performance on specific tasks by learning transferable features from other tasks. It acts as a shared knowledge pool, capturing features common across tasks or relevant to the cell line through a single fully connected layer, followed by a ReLU activation and a Dropout layer. Meanwhile, each task-specific MIL head functions as a task expert, focusing on precise classification for its respective task.

3.3.2 Task groups

We have data from 200 tasks in total, but our primary focus is on the performance of 6 specific tasks. While we aim to utilize as much data as possible, incorporating too much noisy data could degrade performance. To balance

this, we designed 10 groups of potential task combinations and conducted preliminary experiments across these groups to determine the optimal combination for further experiments. A summary is shown on Table 3.1.

Name	# Tasks	# Wells	# Compounds	Notes
Vanilla	6	14,093	6,241	Important tasks
Basic (65)	115	81,610	40,291	AUC $\geq$ 0.65
Basic (70)	83	81,195	40,065	AUC $\geq$ 0.70
Basic (75)	51	80,199	39,498	AUC $\geq$ 0.75
Order 1 (65)	101	79,150	38,677	order 1 $\cap$ AUC $\geq$ 0.65
Order 1 (75)	28	72,740	34,513	order 1 $\cap$ AUC $\geq$ 0.75
Order 2 (65)	130	81,250	39,894	order 2 $\cap$ AUC $\geq$ 0.65
Order 2 (75)	36	73,409	34,766	order 2 $\cap$ AUC $\geq$ 0.75
Order 3 (65)	138	81,465	40,008	order 3 $\cap$ AUC $\geq$ 0.65
Superset	200	84,030	41,843	All tasks

Table 3.1: Task groups' names and details.

The grouping criteria are based on task performance and gene-to-gene relationships between tasks. Using the STL models, we assess individual task performance through metrics such as AUC, balanced accuracy, and F1-score. If a task performs suboptimally, it suggests that the task is difficult to classify in our current setup and may contribute little to a unified model. To address this, we filter tasks by selecting those with an AUC of 0.65 or higher in the STL model.

Additionally, we prioritize tasks that are biologically related to our 6 key tasks. To achieve this, we use gene-to-gene relationships from the STRING database[1], a resource for known and calculated protein-protein interactions. We classify gene pairs based on their interaction relationships: order 1 if they have a direct interaction, order 2 if they are connected through a third gene, and order 3 if they are connected via a chain of two intermediary genes.

3.3.3 Batch processing

Compared to STL, MTL increases both the amount of data and the number of trainable parameters, primarily due to the addition of multiple classification heads. This results in a significant rise in training time. While training a single-task model typically takes less than a few minutes per epoch, an epoch for the Superset model can take several hours. The most time-intensive aspects are the computation of model calculations (forward pass), gradient calculations (backward pass), and parameter updates (optimization step). Although we implemented gradient accumulation to reduce the frequency of

[1]https://string-db.org

parameter updates without altering the model, the other processes remain slow, and the overall training time is still lengthy.

The most effective way to accelerate the process would be to implement batch processing. However, due to the nature of MIL and our model design, the optimal approach is to use a batch size of 1 to accommodate varying sizes of compound bags. In our setup, where training occurs at the well level, over 98% of the data have identical dimensions, with less than 2% missing only one FOV. To expedite model training, we compromised slightly on data integrity by augmenting these outliers with linear interpolation, enabling us to run more experiments efficiently. A small comparison of training times is presented in Table 3.2, where we ultimately settled on a batch size of 256 for model training.

Batch size	Training time/epoch
64	340s
128	173s
256	91s

Table 3.2: Training time per epoch for Superset model.

3.3.4 AMTL Task Weights

To better balance the training progress of tasks with varying characteristics, we adopt the AMTL method proposed by [18], which is formulated as

$$w_t = \left(1 - \frac{\overline{Acc_t}}{m \cdot p_t}\right)^{\gamma}$$
$$L_{total} = \prod_{t=1}^{N_T} L_t^{w_t}.$$

where w_t represents the task weight, $\overline{Acc_t}$ is the current task accuracy, m is a margin, p_t is the task potential, and γ is a scaling factor. To compute the total loss, they express the multi-task loss as a weighted geometric mean of the individual task losses.

Building on this baseline, we adapt the approach by concentrating on our 6 key tasks and halting when the desired performance is reached. The modified

formula is presented as follows:

$$w_t = \left| 1 - \frac{\overline{AUC_t}}{\max(m \cdot p_t, p)} \right|_+^\gamma$$
$$L_{total} = \frac{1}{N_T} \sum_{t=1}^{N_T} w_t \cdot L_t.$$

where $|x|_+ = \max(x, 0)$, $\overline{AUC_t}$ is the current task AUC, and p is a fixed potential. In our setup, we prioritize achieving an AUC of 0.85 across all tasks over having a single outstanding task. We apply a hard cut-off at $p = 0.85$, meaning when this threshold is reached, w_t becomes 0 for that epoch, preventing updates to the model weights from that task.

This adjustment forces the model to prioritize key tasks, with γ set to 2 and m to 1.1. For calculating the total loss, we opted for the arithmetic mean, as it offers a more practical approach by smoothing out the impact of any underperforming tasks. This allows the model to improve overall performance without excessively penalizing outlier tasks.

3.3.5 Prior loss

We further implement weakly supervised task affinity using data from the STRING database, where combined scores are assigned to pairs of genes. A score of 1 indicates a higher probability that the two genes share similar biological functions, while a score of 0 suggests a lower likelihood. For two tasks, t_i and t_j, we hypothesize that if their combined score, c_{ij}, is high, their well embeddings, $e_{i(1\times D)}$ and $e_{j(1\times D)}$, should be similar, resulting in a high inner product score. In other words, the product $e_i e_j^{tr}$ is expected to show a positive correlation with c_{ij}.

For each well, we extract the feature embeddings after attention aggregation and before the classification module, which is represented as:

$$e_{(T\times D)} = \begin{bmatrix} e_1 \\ e_2 \\ ... \\ e_T \end{bmatrix}$$

where T is the number of tasks and D the embedding dimension. We define the basic prior loss as follows, using the mean square error (*mse*) function:

$$
\begin{aligned}
C &= \begin{bmatrix} c_{11} & c_{12} & ... & c_{1T} \\ c_{21} & c_{22} & ... & c_{2T} \\ ... & ... & ... & ... \\ c_{T1} & c_{T2} & ... & c_{TT} \end{bmatrix} \\
E &= ee^{tr} \\
&= \begin{bmatrix} e_1 \\ e_2 \\ ... \\ e_T \end{bmatrix} \begin{bmatrix} e_1^{tr} & e_2^{tr} & ... & e_T^{tr} \end{bmatrix} \\
&= \begin{bmatrix} e_1e_1^{tr} & e_1e_2^{tr} & ... & e_1e_T^{tr} \\ e_2e_1^{tr} & e_2e_2^{tr} & ... & e_2e_T^{tr} \\ ... & ... & ... & ... \\ e_Te_1^{tr} & e_Te_2^{tr} & ... & e_Te_T^{tr} \end{bmatrix} \\
&= E^{tr} \\
Loss_{prior} &= mse(C - E) \\
&= \frac{1}{T^2} \sum_{i}^{T} \sum_{j}^{T} (c_{ij} - e_i e_j^{tr})^2 \\
Loss_{total} &= Loss_{cls} + \lambda_{prior} Loss_{prior}
\end{aligned}
$$

λ is a hyperparameters chosen from [0.001, 0.01, 0.1, 1, 5]. Moreover, since we are only concerned with the results of key tasks, we can further add task' important value v_i^p in our loss functions as follows:

$$
Loss_{prior} = \frac{1}{T^2} \sum_{i}^{T} \sum_{j}^{T} v_i^p (c_{ij} - e_i e_j^{tr})^2
$$

where v_i^p is defined as:

$$
v_i^p = \begin{cases} 0.9, & \text{if } i \in K \\ 0.1, & \text{otherwise} \end{cases}
$$

with K being the set of key task indices, where $|K| = 6$ in our case.

3.3.6 Pseudo labels

To address the issue of label deficiency, with only 4% of the data being labeled, we implement the pseudo-labeling technique. This approach leverages the large amount of unlabeled data, expanding the dataset size and improving the model's ability to generalize. The algorithm is described at Algorithm 1.

Algorithm 1: Pseudo Labels

Input: Dataset $D = (D_{train}, D_{valid}, D_{test})$, label $L = (L_{train}, L_{valid}, L_{test})$, warm up epoch e^{warm}, validation frequency f^{valid}, pseudo label update frequency f^{pseudo}, total epoch E

Variables: Model M, Pseudo labels L', epoch e, thresholds θ, is_update_pseudo

Functions: update_pseudo($\cdot$), calculate_thres($\cdot$)

Initialization: M initializes, $e \leftarrow 0$, $L' \leftarrow L_{train}$, is_update_pseudo = False

while $e < e^{warm}$ **do**
  Train $M(D_{train}, L')$;
  if e % $f^{valid} = 0$ **then**
    Validate $M(D_{valid}, L_{valid})$
  $e \leftarrow e + 1$
while $e < E$ **do**
  if *is_update_pseudo* **then**
    L' $\leftarrow$ update_pseudo($D_{train}, L_{train}, \theta, M$);
    is_update_pseudo $\leftarrow$ False
  Train $M(D_{train}, L')$;
  if e % $f^{valid} = 0$ **then**
    Validate $M(D_{valid}, L_{valid})$
  if e % $f^{pseudo} = 0$ **then**
    $\theta \leftarrow$ calculate_thres(D_{valid}, L_{valid}, M);
    is_update_pseudo $\leftarrow$ True
  $e \leftarrow e + 1$
Test $M(D_{test}, L_{test})$

Inside Algorithm 1, the function $update\_pseudo(\cdot)$ is responsible for updating the new pseudo labels for the training set, while the function $calculate\_thres(\cdot)$ recalculates the selection thresholds from the validation set. The details of these functions are further outlined in Algorithm 2 and Algorithm 3.

To emphasize the key tasks, we introduce two parameters, α^{pos} and α^{neg}. Setting $\alpha^{pos} > 1$ reduces the number of positive pseudo labels from other tasks, while $\alpha^{neg} < 1$ decreases the number of negative pseudo labels. The updated function $update\_pseudo_emphasis$ can be found at Algorithm 4.

Algorithm 2: Update_Pseudo

Input: Dataset D_{train}, label L_{train}, thresholds θ, model M
Functions: $L_{train}(x)$ returns the label of x, $M(x)$ returns the model classification of x, $(\cdot)_t$ returns the corresponding value for task t
Initialization: $L' \leftarrow L_{train}$

```
for t in [1, ..., T] do
    for x ∈ D_train do
        if L_train(x)_t is nan then
            if M(x)_t ≥ θ_t^pos then
                L'(x)_t = 1
            if M(x)_t ≤ θ_t^neg then
                L'(x)_t = 0
```

Algorithm 3: Calculate_Thres

Input: Dataset D_{valid}, label L_{valid}, model M
Variables: Thresholds $\theta = (\theta_1^{pos}, \theta_1^{neg}, \theta_2^{pos}, \theta_2^{neg}, ..., \theta_T^{pos}, \theta_T^{neg})$, pos, neg
Functions: $L_{valid}(x)$ returns the label of x, $M(x)$ returns the model classification of x, $(\cdot)_t$ returns the corresponding value for task t

```
for t in [1, ..., T] do
    pos ← [];
    neg ← [];
    for x ∈ D_train do
        if L_valid(x)_t = 1 and M(x)_t ≥ 0.5 then
            pos.append(M(x)_t)
        if L_valid(x)_t = 0 and M(x)_t < 0.5 then
            neg.append(M(x)_t)
    θ_t^pos ← 90^th percentile of pos;
    θ_t^neg ← 10^th percentile of neg;
```

Algorithm 4: Update_Pseudo_Emphasis

Input: Dataset D_{train}, label L_{train}, thresholds θ, model M, tasks emphasis α^{pos} and α^{neg}, important tasks K

Functions: $L_{train}(x)$ returns the label of x, $M(x)$ returns the model classification of x, $(\cdot)_t$ returns the corresponding value for task t

Initialization: $L' \leftarrow L_{train}$

for $t \in [1, 2, ..., T]$ **do**
  for $x \in D_{train}$ **do**
    if $L_{train}(x)_t$ *is nan* **then**
      if $t \in K$ **then**
        if $M(x)_t \geq \theta_t^{pos}$ **then**
          $L'(x)_t = 1$
        if $M(x)_t \leq \theta_t^{neg}$ **then**
          $L'(x)_t = 0$
      else
        if $M(x)_t \geq \alpha^{pos}\theta_t^{pos}$ **then**
          $L'(x)_t = 1$
        if $M(x)_t \leq \alpha^{neg}\theta_t^{neg}$ **then**
          $L'(x)_t = 0$

Chapter 4

Results

4.1 Baseline model

In this section, we present the performance of the baseline models. The first model utilizes features from Cell Profiler (Cell Profiler) with an MLP classifier, while the second model replaces these features with those from self-trained foundation models, specifically DINOv2. Given the significant advantage observed with DINOv2, all subsequent experiments will exclusively use DINOv2 features.

Task/Model	Cell Profiler → MLP	DINOv2 → MLP
T1	0.642	0. 671
T2	0.670	0.664
T3	0.757	0. 748
T4	0.683	0.750
T5	0.568	0.621
T6	0.700	0.698
Average	0.670	0.692

Table 4.1: Two baseline model AUC results on key 6 tasks.

4.2 Hypterparameters searching

We first conducted explicit experiments on T1, evaluating the performance of various models across a comprehensive range of hyperparameters. In the following paragraphs, we discuss the effects of each parameter individually by calculating the marginal results, allowing us to better understand their respective impacts. Overall, we tested 2 different models, 5 model sizes, 5 learning rates, 4 crop numbers, and 2 activation functions. The results of these evaluations are presented in Table 4.2 and explained as follows.

Model	
Name	AUC
ABMIL	0.702
DSMIL	0.710

Model size	
# Params	AUC
13,154	0.647
25,666	0.687
51,074	0.710
103,426	0.707
214,274	0.701

Table 4.2: STL results with different attention-based MIL models for T1.

MIL Models Both ABMIL and DSMIL improve classification performance compared to MLPs. However, DSMIL demonstrates greater potential in our experiments. Therefore, we choose to focus exclusively on DSMIL for future work.

Model Sizes Among the five different model sizes tested, the model with approximately 50,000 trainable parameters provides the most consistent advantages in terms of performance and efficiency.

Learning Rates We find that learning rates equal to or greater than 1e-4 cause instability during training, while those below 1e-6 are inefficient. As a result, we settle on using learning rates of 5e-5, 1e-5, 5e-6, and 1e-6 for further experiments.

Crop Numbers and Activation Functions There are no significant or consistent differences in performance based on crop numbers or activation functions. Ultimately, we decide to retain both crop numbers 4 and 16, as well as the ReLU activation function for future trials.

Task/Model	DINOv2 $\rightarrow$ MLP	DINOv2 $\rightarrow$ MIL
T1	0.671	0.710
T2	0.664	0.691
T3	0.748	0.764
T4	0.750	0.769
T5	0.621	0.626
T6	0.698	0.710
Average	0.692	0.712

Table 4.3: Two baseline model AUC results on key 6 tasks.

4.3 STL Classification

After fixing hypterparameter space, we run experiments on all the 200 tasks, where we run explicit experiments on other 5 key tasks, and 2 experiments on other 194 tasks. We need the single task results for all of the tasks so that we have the potential performance for prior loss method. The key tasks results are shown in Table 4.3.

4.4 MTL grouping selection

After conducting explicit experiments, the results presented in Table 4.4 indicate that the intersection of the "Order 1" group, which is biologically closer to our key tasks, and the "Basic 75" group, which contains more potentially learnable tasks, yields the best performance. Therefore, in future multi-task learning experiments, we will focus on this group and report results based on its performance.

Group	T1	T2	T3	T4	T5	T6	Average
Vanilla	0.719	0.688	0.771	0.773	0.634	0.709	0.715
Basic (65)	0.720	0.708	0.740	0.783	0.656	0.730	0.723
Basic (70)	0.715	0.704	0.738	0.776	0.672	0.734	0.723
Basic (75)	0.720	0.710	0.764	0.774	0.656	0.712	0.723
Order 1 (65)	0.718	0.710	0.749	0.773	0.670	0.723	0.724
Order 1 (75)	0.732	0.712	0.776	0.773	0.664	0.717	0.729
Order 2 (65)	0.711	0.717	0.771	0.780	0.639	0.728	0.724
Order 2 (75)	0.714	0.717	0.768	0.779	0.644	0.719	0.724
Order 3 (65)	0.727	0.719	0.735	0.787	0.656	0.728	0.725
Superset	0.715	0.739	0.752	0.772	0.663	0.726	0.728

Table 4.4: MTL AUC results with grouping strategies.

4.5 MTL Classification

Table 4.5 presents the results of MTL in comparison with the previously mentioned STL. From our current setup, it is shown that the original multi-task MIL learning demonstrates an advantage over the addition of AMTL, Prior loss, and Pseudo loss.

AMTL The task weights of the key tasks show a slight decrease after implementing AMTL, indicating an improvement in the validation AUCs. While we are confident in the potential of AMTL, it introduces several challenges during implementation. One issue arises after the initial warm-up phase of training: when we multiply the loss by the task weights, we observe

	Method (Feature → Model)		T1	T2	T3	T4	T5	T6	Average
STL	Cell Profiler	MLP	0.642	0.670	0.757	0.683	0.568	0.700	0.670
	DINOv2 → Mean	MLP	0.671	0.664	0.748	0.750	0.621	0.698	0.692
	DINOv2	MIL	0.710	0.691	0.764	0.769	0.626	0.710	0.712
MTL	DINOv2 → Mean	MLP	0.697	0.655	0.762	0.749	0.629	0.717	0.702
	DINOv2	$\text{MIL}_{vanilla}$	0.719	0.688	0.771	0.773	0.634	0.709	0.715
		MIL	0.732	0.712	0.776	0.773	0.664	0.717	0.729
		MIL + AMTL	0.712	0.706	779	0.770	0.649	0.717	0.722
		MIL + Prior	0.738	0.694	0.775	0.760	0.627	0.726	0.720
		MIL + Pseudo	0.715	0.707	0.773	0.761	0.661	0.721	0.723

Table 4.5: All AUC results with different methods.

a performance drop, despite trying different normalization techniques. This suggests that tuning the learning rates becomes more complex in the presence of AMTL. Additionally, determining the optimal margin to use and how to define the cut-off threshold have proven to be crucial hyperparameters, requiring extensive experimentation. These challenges highlight the need for careful parameter tuning to fully unlock the benefits of AMTL. The task weights and losses are shown in Figure 4.1.

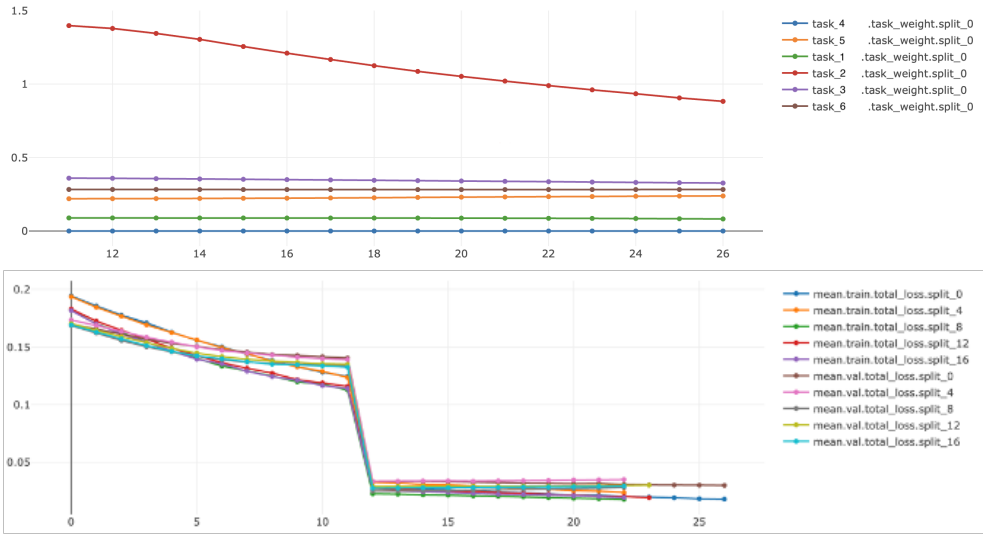

Figure 4.1: AMTL task weights and losses.

Prior loss The prior loss and total loss both show a decreasing trend on the validation set, while the AUC improves, as illustrated in Figure 4.2. This provides confidence that the prior assumption we implemented is reasonable. However, there remains room for refinement and improvement. Several

aspects still require further investigation, such as whether the combined score from the STRING database is the most appropriate label for this context and what constitutes an optimal balance between the prior loss and the classification loss. Addressing these questions will be essential in further enhancing the model's performance and ensuring that prior knowledge is effectively integrated into the learning process.

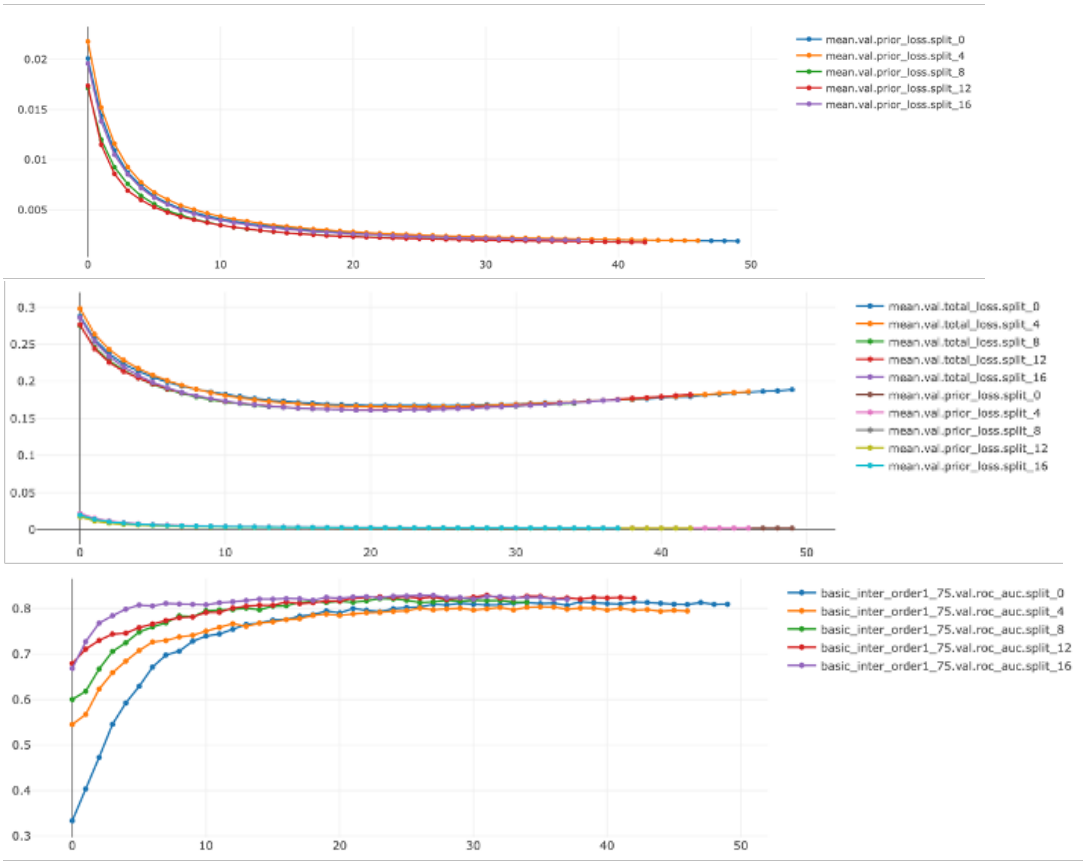

Figure 4.2: Prior loss. (1)Validation prior loss (2)Validation prior loss along with total loss (3)Validation AUC

Pseudo loss From Figure 4.3, we can observe that both the pseudo labels and the original labels are learnable, although the pseudo-labeled data contribute significantly more to the overall loss. This discrepancy is likely due to the noisiness inherent in the pseudo labels. After the warm-up epoch, we notice that the validation loss increases once pseudo labels are introduced, which may suggest that the noise in the pseudo labels affects the model's ability to generalize effectively.

Additionally, we evaluated the thresholds used for key tasks, as shown in Figure 4.3. Here, approximately 10% to 20% of the original labeled data were re-labeled as pseudo labels. This ratio aligns well with our choice of using the 90th percentile as the threshold. A higher percentage would indicate that the model is overly confident in its calculations, while a lower percentage would suggest underconfidence. This balance gives us some assurance that the pseudo-labeling process is operating within a reasonable confidence range, but further refinement is still necessary to reduce noise and improve model stability.

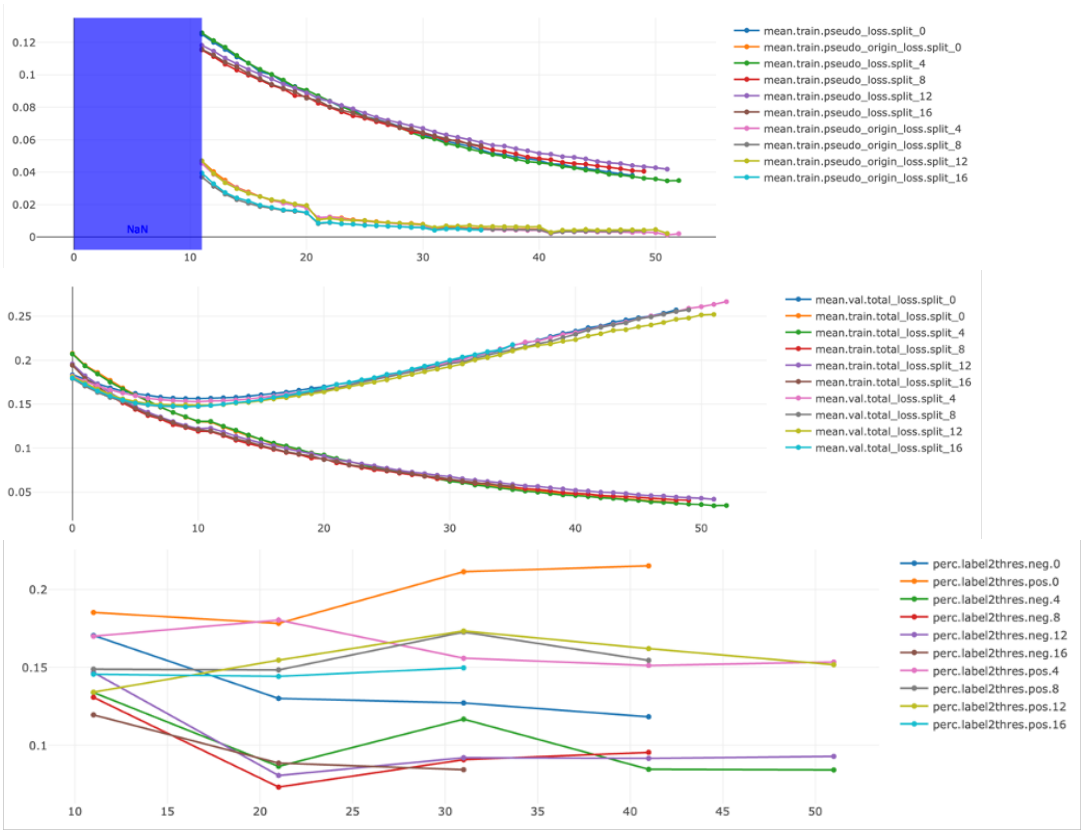

Figure 4.3: Pseudo loss. (1)Pseudo_loss: losses of pseudo labeled data. Pseudo_origin_loss: losses of originally labeled data. (2)Ratio of original labels that serve thresholds.

Chapter 5

Conclusion

In this thesis, we explored the use of attention-based MIL for both single-task and multi-task learning in HCI for drug discovery. Our findings highlight several key insights regarding the effectiveness of these methods in handling heterogeneous data and multiple tasks simultaneously.

First, for single-task classification, we observed that attention-based MIL outperformed traditional MLP models. This improvement is largely due to the heterogeneous nature of our data, where the final classification decision often depends on just a few key instances within a bag of observations, that is, a few FOVs from the same compound. The attention mechanism enables the model to focus on these critical instances, leading to more accurate assessments and making MIL a more suitable approach for handling diverse biological data.

Building on this success, we extended the model to a multi-task learning setup, which significantly enhanced overall performance by introducing a shared backbone that allows for better feature learning across tasks. However, selecting the right group of tasks to train together proved crucial. Through our experiments, we found that the best strategy involves balancing biological relevance with the model's learnability. Specifically, forming task groups based on both biological relationships and the model's ability to learn from these tasks yielded the most promising results. This dual consideration ensures that the shared representation can generalize well across related tasks while still capturing the unique characteristics of each.

In an effort to further improve the multi-task learning framework, we implemented several advanced techniques, including task-weighting systems, prior loss, and pseudo labeling. While these modifications did not lead to significant performance improvements in our current setup, they provided valuable insights into the complexities of multi-task learning. We believe that these techniques hold potential for enhancing model performance, and our

future work will focus on refining their implementation to better leverage their benefits. Understanding how to integrate such methods more effectively will be key to pushing the boundaries of multi-task learning in HCI .

In conclusion, attention-based MIL offers clear advantages for single-task learning in heterogeneous datasets, and extending this approach to multi-task learning can further enhance model performance. However, the challenge of selecting and grouping tasks remains critical, and while additional modifications such as task-weighting and prior loss show potential, their full impact is yet to be realized. These findings set the stage for future exploration into more refined multi-task learning strategies that can drive even greater advancements in drug discovery applications.

Bibliography

[1] D. M. J. Ando, Cory Y. McLean, and Marc Berndl. Improving phenotypic measurements in high-content imaging screens. *bioRxiv*, 2017.

[2] Heba Askr, Enas Elgeldawi, Heba Aboul Ella, Yaseen AMM Elshaier, Mamdouh M Gomaa, and Aboul Ella Hassanien. Deep learning in drug discovery: an integrative review and future challenges. *Artificial Intelligence Review*, 56(7):5975–6037, 2023.

[3] Denise B Catacutan, Jeremie Alexander, Autumn Arnold, and Jonathan M Stokes. Machine learning in preclinical drug discovery. *Nature Chemical Biology*, pages 1–14, 2024.

[4] Christophe Zimmer Daniel Krentzel, Spencer L. Shorte. Deep learning in image-based phenotypic drug discovery. pages 538–554,, 2023.

[5] Jan P Engelmann, Alessandro Palma, Jakub M Tomczak, Fabian Theis, and Francesco Paolo Casale. Mixed models with multiple instance learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3664–3672. PMLR, 2024.

[6] Samman Fatima, Sikandar Ali, and Hee-Cheol Kim. A comprehensive review on multiple instance learning. *Electronics*, 12(20):4323, 2023.

[7] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[8] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning, 2018.

[9] Héctor Corrada Bravo David Simcha Benjamin Langmead W. Evan Johnson Donald Geman Keith Baggerly Rafael A. Irizarry Jeffrey T. Leek,

Robert B. Scharpf. Tackling the widespread and critical impact of batch effects in high-throughput data. *Nat Rev Genet 11*, 2010.

[10] W Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127, 2007.

[11] Bin Li, Yin Li, and Kevin W Eliceiri. Dual-stream multiple instance learning network for whole slide image classification with self-supervised contrastive learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14318–14328, 2021.

[12] Ming Y Lu, Drew FK Williamson, Tiffany Y Chen, Richard J Chen, Matteo Barbieri, and Faisal Mahmood. Data-efficient and weakly supervised computational pathology on whole-slide images. *Nature biomedical engineering*, 5(6):555–570, 2021.

[13] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. 2024.

[14] Rizwan Qureshi, Muhammad Irfan, Taimoor Muzaffar Gondal, Sheheryar Khan, Jia Wu, Muhammad Usman Hadi, John Heymach, Xiuning Le, Hong Yan, and Tanvir Alam. Ai in drug discovery and its clinical relevance. *Heliyon*, 9(7), 2023.

[15] Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017.

[16] Srijit Seal, Maria-Anna Trapotsi, Ola Spjuth, Shantanu Singh, Jordi Carreras-Puigvert, Nigel Greene, Andreas Bender, and Anne E Carpenter. A decade in a systematic review: The evolution and impact of cell painting. *ArXiv*, 2024.

[17] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

[18] Hayoung Yun and Hanjoo Cho. Achievement-based training progress balancing for multi-task learning. pages 16935–16944, October 2023.

[19] Yan Zhao, Xiuying Wang, Tongtong Che, Guoqing Bao, and Shuyu Li. Multi-task deep learning for medical image computing and analysis: A review. *Computers in Biology and Medicine*, 153:106496, 2023.