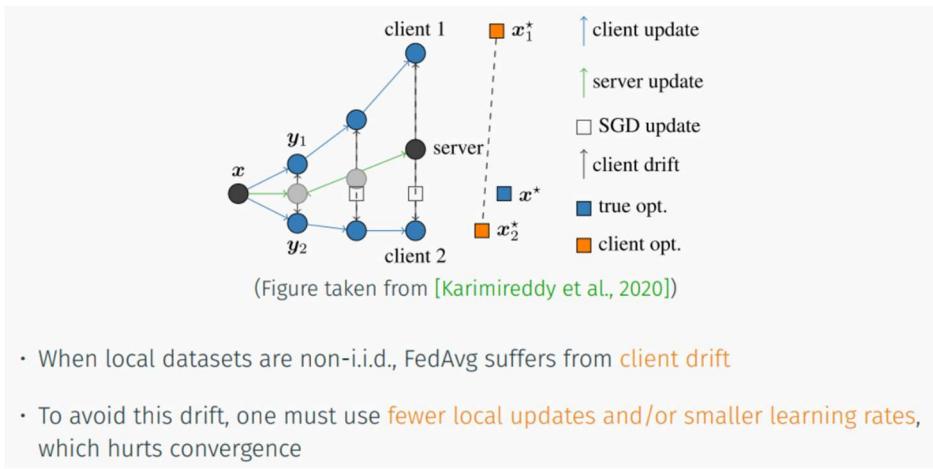


# به نام خدا

تمرین سری پنجم  
درس تحلیل هوشمند تصاویر زیست پزشکی  
دکتر محمد حسین رهبان

فرزان رحمانی  
۴۰۳۲۱۰۷۲۵

سوال اول  
(آ)



- When local datasets are non-i.i.d., FedAvg suffers from **client drift**
- To avoid this drift, one must use **fewer local updates and/or smaller learning rates**, which hurts convergence

الگوریتم FedAvg به میانگین‌گیری پارامترهای مدل‌های آموزش‌دیده محلی در بین client‌ها (locally trained models across clients) متکی است. هنگامی که داده‌ها غیر هeterogeneous (non-IID) هستند، مدل‌های محلی آموزش داده شده توسط client‌های مختلف به طور قابل توجهی واگرا (diverge significantly) می‌شوند زیرا توزیع داده‌های آموزشی آنها بسیار متفاوت است.

در واقع این الگوریتم به دلایل زیر دچار چالش می‌شود:

- Weight Shift: پارامترهای مدل از هر client به طور گستره‌ای متفاوت است، که میانگین‌گیری ساده را در ایجاد یک global مدل به خوبی تعمیم می‌دهد، بی اثر می‌کند. به عبارت دیگر، وزن مدل از client‌های مختلف می‌تواند بدليل داده‌های ناهمگن به طور قابل توجهی تغییر کند، که باعث می‌شود میانگین‌گیری ساده پارامترها در ارائه مدل global کمتر مؤثر باشد.

- Overfitting و Poor Generalization: در مدل‌های محلی (local) مدل‌های local احتمالاً به توزیع داده‌های مربوطه خود overfit می‌شوند و توانایی آنها برای بازنمایی الگوهای global را کاهش می‌دهند. به عبارت دیگر، مدل‌های local آموزش‌دیده بر روی داده‌های غیر IID مستعد overfit با توزیع داده‌های خاص خود هستند که در نتیجه تعمیم ضعیفی به توزیع داده‌های global ایجاد می‌کند.

- همگرایی آهسته تر: داده‌های غیر IID اغلب به دورهای ارتباطی بیشتری (more communication rounds) نیاز دارند تا FedAvg همگرا شود، که زمان و هزینه های آموزشی را افزایش داده و فرآیند را ناکارآمدتر می‌کند. واگرایی مدل محلی (local): در سناریوهای غیر IID، مجموعه داده‌های محلی در هر client می‌تواند توزیع های متفاوتی داشته باشد. این امر باعث می‌شود که مدل‌های محلی آموزش‌دیده شده بر روی این مجموعه داده‌ها از هم diverge شوند، که منجر به بهروزرسانی‌های متناقض در هنگام میانگین‌گیری در سرور می‌شود.

In this work, we mainly focus on the heterogeneous data or what is known as data non-independently and identically distributed (non-iid) problem in FL [22]–[24]. Most FL algorithms are based on the FedAvg [1], where the clients train a local model, then upload them to the server and get a global model by averaging the client model's parameters. But under the non-iid scenarios, these algorithms require more communication rounds to converge and result in poor performance.

There have been some studies trying to address the non-iid issue, which can be divided into three types. (1) **Weight-**

**constraint methods** [25]–[32] use regularization, normalization, or other approaches to constrain the parameters of the client's local models, which makes the local training more stable and reduces the bias of the local models. (2) **Client-selection methods** [33]–[37] optimise the client selection strategy by designing specific metrics. (3) **Knowledge distillation methods** [38]–[46] use knowledge distillation to reduce the impact of the non-iid data. However, most methods only optimize the client model and just constrain the parameters of the local model in some naive ways, e.g., regularization and distillation. Therefore, they can only achieve minor improvement in the final model performance.

Observing the challenge in the non-iid scenarios and the limitations of the prior work, in this work, we propose a federated learning algorithm with global and local distillation. We design two optimization modules in our FL algorithm: the noise-distillation for the server model and the self-distillation

(ب)

#### Self-Distillation on the Client Side

برای پاسخ به این سوال به شکل زیر از مقاله توجه کنید که به خوبی توضیح داده سپس توضیحات من آمده است:

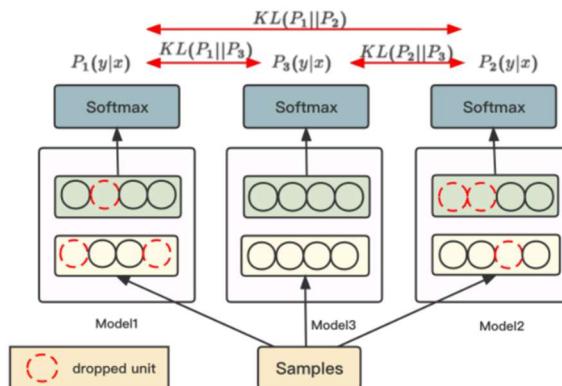


Fig. 3. Self Distillation: The client model is composed of three sub-models with the same structure. Two of them are different in the parameters of the dropout layer, while the other model is a model that has been trained in the previous epoch. Samples were simultaneously trained on three sub-models and distilled by KL loss.

مکانیسم تقویت آموزش local: تقطیر خود (Self-distillation) از لایه‌های dropout در مدل محلی (local model) استفاده می‌کند تا پیش‌بینی‌های کمی متفاوت (slightly different predictions) برای ورودی یکسان ایجاد کند. سپس این خروجی‌ها با استفاده از KL divergence برای بهبود سازگاری و استحکام (consistency and robustness) تراز می‌شوند. علاوه بر این، خروجی‌های مدل فعلی (current model's outputs) با خروجی‌های مدل دوره قبل (previous epoch's model) برای تثبیت آموزش و جلوگیری از excessive drift، تراز می‌شوند.

در واقع، سه نوع ضرر را در طول آموزش محاسبه می‌کند:

- ضرر بین خروجی‌های مدل و برجسب‌های واقعی.
- ضرر بین خروجی‌های به دست آمده با تنظیمات مختلف dropout (برای بهبود استحکام).
- ضرر بین خروجی‌های مدل فعلی و خروجی‌های مدل قبلی (برای حفظ ثبات).

مزایای آن یادگیری فدرال:

by forcing the model to generalize better Overfitting (across variations in the input data

- با هموارسازی پیش‌بینی‌ها در طول زمان و به حداقل رساندن تغییرات ناگهانی، مدل محلی را به تعمیم بهتر تشویق می‌کند. (preventing the local model from drifting too far from the global model)
  - با ترکیب بهروزرسانی‌های محلی سازگارتر (more consistent local updates)، استحکام را به ویژه در های غیر IID بپسود می‌بخشد.
- در مقاله هم در این قسمت‌ها به آنها اشاره شده است:

### C. Self Distillation

In federated learning, the clients' data is usually insufficient and unbalanced (especially for non-iid data), and the model trained with it is likely to be overfitted, thus affecting the global model performance. Knowledge distillation can mitigate overfitting, but distillation methods based on teacher-student models require specific designs for the teacher model.

We note that if the mechanism of the dropout layer is utilized, distillation can be accomplished using only one model. The outputs of the same input sample will be different when it passes through a model with dropout layers at different times. And we can distill the model by reducing the distance of these outputs, which omits the teacher model in traditional knowledge distillation and alleviates the overfitting problem.

Therefore, as shown in Figure 3, we have designed a method for local training in the client called self-distillation. Specifically, let us take the classification task as an example, the model parameters for the current round are  $w_t$ , the training samples are  $(x, y)$ , and the model output is represented by the function  $f$ . We make a copy of the initial model  $M_0$  at each round of local training and set it untrainable. And the method is constrained by three loss functions. The first is the loss of model outputs to the true labels like:

$$L_1 = CE(f_1(w_t; x), y) + CE(f_2(w_t; x), y). \quad (3)$$

Here  $CE(\cdot, \cdot)$  means the commonly-used cross-entropy loss. The probability distributions  $f_1(w_t; x)$  and  $f_2(w_t; x)$  are obtained by feeding the samples into the model (with dropout) twice. Due to the dropout layers in the network,  $f_1$  is different from  $f_2$ . The second loss function is defined as follows:

$$L_2 = KL(f_1(w_t; x) || f_2(w_t; x)), \quad (4)$$

which is to compute the KL divergence between their probability distributions. Then the samples are passed through the previously fixed model  $M_0$  to obtain the probability distribution  $f_3$ . Finally, we compute the distance between the outputs of the current model and the previous model by

$$L_3 = KL(f_1(w_t; x) || f_3(w_t; x)) + KL(f_2(w_t; x) || f_3(w_t; x)). \quad (5)$$

The total loss is:

$$L = \alpha L_1 + \beta L_2 + \gamma L_3, \quad (6)$$

where the  $\alpha, \beta, \gamma$  are hyperparameters.

The  $L_2$  loss improves the robustness of the local model and reduces the risk of overfitting, while the  $L_3$  loss prevents the local model from drifting too far from the previous global model and stabilizes the training process.

نحوه تولید و مزیت‌های Noise Samples

برای پاسخ به این سوال به شکل زیر از مقاله توجه کنید که به خوبی توضیح داده سپس توضیحات من آمده است:

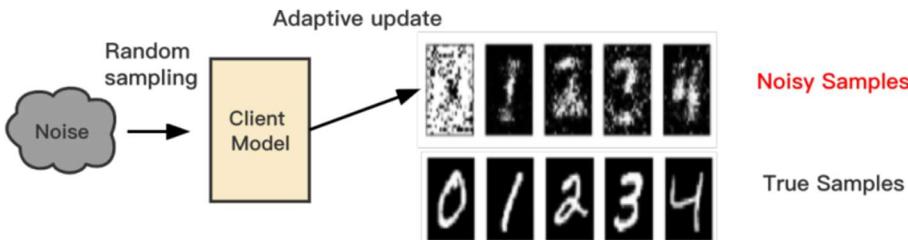


Fig. 4. Noise Generation: First, the noise is sampled from the random distribution and output through the client model as a train sample. Then, the constructed loss function  $L_e$  is used for reverse updating to obtain a noisy sample that approximates the real sample.

نمونه‌های نویز چگونه تولید می‌شوند:

- Noise samples are initialized randomly using a confidence-based (normal distribution) algorithm (adapted). This algorithm adapts the client's model to the noisy samples by minimizing a confidence loss (terms of their impact on model outputs).
  - Yadگیری فدرال به دلیل محدودیت های حریم خصوصی، اشتراک گذاری داده های واقعی را در بین client ها منع می کند. نمونه های نویز، دانش مشترک (shared knowledge) را بدون افشای داده های حساس واقعی مزایای این نمونه ها در Yadگیری فدرال:
  - انتقال دانش بین مشترک را تسهیل می کند.
  - با استفاده از شبیه داده ها (pseudo-data) برای تراز کردن خروجی ها در بین مدل ها، اختلاف بین مدل های client را کاهش می دهد (weight shift) (models).
  - تعیین مدل global را با ارائه pseudo-representation از توزیع های داده های متنوع افزایش می دهد.
  - سربار communication rounds و محاسبات را کاهش می دهد.
- در مقاله هم در این قسمت ها به آنها اشاره شده است: and use the pseudo-sample's output probability distribution

$$\begin{aligned}\hat{y}_1 &= f_{M_1}(w_1, \hat{x}_1), \\ \hat{y}_2 &= f_{M_2}(w_2, \hat{x}_2),\end{aligned}\quad (8)$$

as the soft label information for the noisy samples.

Although random noisy samples can already distill the model, it is possible that the probability distribution of random noise passing through the model output tends to be uniformly distributed. The model does not consider the sample to belong to any category which makes the KL divergence among different client models is small and distillation may be less effective.

We define  $n$  noisy samples with  $h$  dim features as  $\hat{x} \in \mathbb{R}^{n \times h}$  and the task is  $c$  classification. The client model parameters are  $w \in \mathbb{R}^{h \times c}$  and let  $z$  denote the output of the noisy samples after the model.

To attach probabilistic meaning to the model output, a softmax operation is performed on the output  $z$  to obtain the normalized output probability distribution  $p$ . Then we can define the confidence loss function  $L_e$  of the noisy samples for the current model by

$$z = f(w, \hat{x}), \quad p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}, \quad L_e = \sum_{i=1}^c p_i \log p_i. \quad (9)$$

From the above equation, it can be seen that the smaller the confidence loss, the more effective the model distillation is. And we filter out the appropriate noise by setting a suitable threshold. However, due to the large noise space, it is difficult to quickly sample all the samples that satisfy the threshold. Therefore, we consider setting the features of the noisy samples as trainable parameters, and then update the noisy data by deriving the confidence loss function like

$$\hat{x} = \hat{x} - \eta \frac{\delta L_e}{\delta \hat{x}}. \quad (10)$$

Instead of using the adversarial generation network, this method just increases the noisy sample's confidence, the training often requires only a few iterations to obtain suitable noisy samples.

#### D. Noisy sample Generation

The federated training is difficult due to the distributed data. An intuitive idea to augment local training is obtaining samples from other clients, but this is not allowed in federated learning scenarios.

We find that the ultimate goal of data enhancement is to use samples from other clients to make the local model be closer to the others so that when the same sample is passed through different client models, the output probability distribution between them should be as close as possible.

At this point, the specific value of the sample is no longer important, its main role is to reduce the distance between client models. Thus, we propose a method for constructing samples based on random noise which is shown in Figure 4.

Given any two client models  $M_1, M_2$ , we use the normal distribution to sample the noisy data as pseudo-samples by

$$\begin{aligned}\hat{x}_1 &\sim \mathcal{N}(\mu_1, \sigma_1), \\ \hat{x}_2 &\sim \mathcal{N}(\mu_2, \sigma_2),\end{aligned}\quad (7)$$

برای پاسخ به این سوال به شکل زیر از مقاله توجه کنید که به خوبی توضیح داده سپس توضیحات من آمده است:

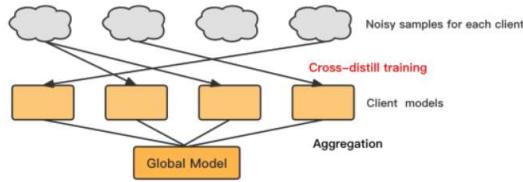


Fig. 5. Noise Distillation: The server will first randomly select the client models participating in distillation, and generate noisy samples for each selected model. Then, the noisy samples are used as training data to distill other client models in a cross method. The noisy sample has the information of its own client model and can play the role of models' parameters normalization.

#### Algorithm 1: Noise Distillation in FedSND

```

Input : Total  $K$  clients,  $N$  client participate in noise distillation, model parameter  $w$ , noisy samples  $(\hat{x}, \hat{y})$ 
Output: global model with parameter  $w_g$ 
1 for  $t \leftarrow 1$  to  $K$  do
2   random sampling  $K_c \in \{1, \dots, K\}$ ,  $|K_c| = N$  ;
3   for  $c \leftarrow 1$  to  $K_c$  do
4     // get the output for noisec
5      $\hat{h}_c = f(w_t, \hat{x}_c)$ ;
6     // distill the current modelt
7      $w_t = \min_{w_t} g(w_t, \hat{h}_c) = KL(\hat{y}_c || \hat{h}_c)$ ;
8   end
9   // average the client model parameters
10   $w_g = \frac{1}{K} \sum_{t=1}^K w_t$ 
11 end
  
```

چگونه مدل های مشتری را تراز می کند:

- هر مدل client نمونه های نویز تولید می کند، که سپس برای fine-tune مدل های client های دیگر در فرآیند نقطیر متقابل (cross-distillation process) استفاده می شود. این تضمین می کند که مدل های client خروجی های مشابهی را برای ورودی های یکسان تولید می کنند.
  - سرور نمونه های نویز را برای هر مدل client تولید می کند و از آنها برای نقطیر متقابل (cross-distillation) بین مدل های client استفاده می کند.
  - این فرآیند توزیع های خروجی مدل های مختلف کلاینت را با کاهش KL divergence بین آنها align می کند.
  - اثربخشی در کاهش Weight-Shift :
  - با تراز کردن خروجی ها در بین مدل ها، مژول noise-distillation تفاوت های پارامترهای مدل ناشی از داده های غیر IID را نormal می کند. این منجر به یک مدل global پایدارتر می شود که به طور مؤثرتری در طول aggregation همگرا می شود.
  - نمونه های نویز به عنوان یک مرجع مشترک عمل می کنند و به نرمال سازی تغییرات در مدل های client ناشی از داده های غیر IID کمک می کنند.
  - نقطیر متقابل تضمین می کند که مدل های client هماهنگ تر هستند و مشکل weight-shift قبل از aggregation را کاهش می دهد.
  - این منجر به یک مدل global پایدارتر و قوی تر می شود که توزیع کلی داده را بهتر نشان می دهد.
- در مقاله هم در این قسمت ها به آنها اشاره شده است:

has already generated corresponding noisy pseudo-samples for different clients, so that it can use those noisy samples to further train the client models.

The main problem faced by the clients is weights-shift, which is caused by the fact that the client does not have access to information about the dataset on other clients.

Our method, as shown in Figure 5, uses noise distillation to overcome this by providing the server with balanced noisy samples to 'correct' client models.

Specifically, noise distillation first uses the noisy samples generated by different client models to distill each other models and then averages the distilled client models to obtain the global model.

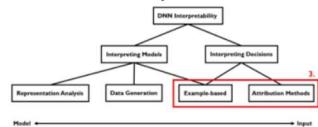
We describe FedSND in algorithm 1. Noise distillation allows noisy samples with high confidence generated by a specific client. Then the client models will have more similar output distributions to others before aggregation. The algorithm helps to reduce the difference among the client model parameters and makes the aggregated global model perform better.

for the client model. **Module1:** In the server, the received client models are usually biased due to the non-iid data, so the naive aggregation method damages the model performance. To overcome this, we design a pseudo-sample generation module in the server and use an adaptive method to update the pseudo-samples to make them more similar to real samples. Since those samples are generated from noise, we call them *noisy samples* for the rest of the paper. We use these samples to update the client models via knowledge distillation and then aggregate the local models to obtain the global model.

## سوال دوم

### Example-based Method .۱

## Types of DNN Interpretability



### Example-based

### Attribution Methods

### Gradient Based

### Backprop. Based

- Which training instance influenced the decision most?
- Still does not **specifically highlight** which features were important

- Influence functions for interpreting black-box methods. **Fragility** of NN model interpretation.

'Sunflower': 59.2% conf.



Influence: 0.09



Influence: 0.14



Influence: 0.42



"Understanding Black-box Predictions via Influence Functions", <https://arxiv.org/pdf/1703.04730.pdf>

"Interpretation of Neural Networks is Fragile", <https://arxiv.org/pdf/1710.10547.pdf>

49

(آ) جزئیات محاسباتی (بخش ۲):

جزیئیات کامل چگونگی محاسبات در صفحه دوم مقاله آمده است. در اینجا به اختصار به توضیح آن می پردازیم. سپس قسمت های مهم مقاله را نیز نشان میدهیم.

چگونگی تغییر پارامترهای یک مدل را در هنگام افزایش وزن (upweight) یا perturb کردن یک داده آموزش اندازه گیری می کند. محاسبات کلیدی عبارتند از:

upweighting a training point .۱: تغییر در پارامترها به دلیل بالا بردن وزن نقطه آموزشی z (Upweighting a Training Point .۱) توسط  $\epsilon$  به صورت زیر مشتق می شود:

$$I_{\text{up,params}}(z) = \frac{d\theta_\epsilon}{d\epsilon} |_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta}),$$

که در آن  $H_{\hat{\theta}}$  هسین empirical risk در  $\hat{\theta}$  است.

Effect on Test Loss .۲: تأثیر بر ضرر در نقطه تست  $z_{\text{test}}$  عبارت است از:

$$I_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta}).$$

این اثر افزایش وزن z را بر test loss محاسبه می کند.

Perturbing Training Inputs .۳: توابع influence همچنین می توانند نحوه تأثیر ویژگی های z بر پیش بینی ها را مدل کنند. اثر تقریبی با استفاده از:

$$I_{\text{pert,loss}}(z, z_{\text{test}}) = -\nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_x \nabla_{\theta} L(z, \hat{\theta}),$$

جایی که  $\nabla_x \nabla_{\theta} L(z, \hat{\theta})$  نحوه تغییر گرادیان loss را با توجه به ویژگی های ورودی نشان می دهد.

این محاسبات کارآمد هستند زیرا با استفاده از تقریب های مرتبه دوم loss landscape از آموزش مجدد چلوگیری می کنند. قسمت های مقاله از مقاله که به تفسیر این موضوع می پردازند:

## 2. Approach

Consider a prediction problem from some input space  $\mathcal{X}$  (e.g., images) to an output space  $\mathcal{Y}$  (e.g., labels). We are given training points  $z_1, \dots, z_n$ , where  $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ . For a point  $z$  and parameters  $\theta \in \Theta$ , let  $L(z, \theta)$  be the loss, and let  $\frac{1}{n} \sum_{i=1}^n L(z_i, \theta)$  be the empirical risk. The empirical risk minimizer is given by  $\hat{\theta} \stackrel{\text{def}}{=} \arg \min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n L(z_i, \theta)$ .<sup>1</sup> Assume that the empirical risk is twice-differentiable and strictly convex in  $\theta$ ; in Section 4 we explore relaxing these assumptions.

### 2.1. Upweighting a training point

Our goal is to understand the effect of training points on a model's predictions. We formalize this goal by asking the counterfactual: how would the model's predictions change if we did not have this training point?

Let us begin by studying the change in model parameters due to removing a point  $z$  from the training set. Formally, this change is  $\hat{\theta}_{-z} - \hat{\theta}$ , where  $\hat{\theta}_{-z} \stackrel{\text{def}}{=} \arg \min_{\theta \in \Theta} \sum_{z_i \neq z} L(z_i, \theta)$ . However, retraining the model for each removed  $z$  is prohibitively slow.

Fortunately, influence functions give us an efficient approximation. The idea is to compute the parameter change if  $z$  were upweighted by some small  $\epsilon$ , giving us new parameters  $\hat{\theta}_{\epsilon,z} \stackrel{\text{def}}{=} \arg \min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n L(z_i, \theta) + \epsilon L(z, \theta)$ . A classic result (Cook & Weisberg, 1982) tells us that the influence of upweighting  $z$  on the parameters  $\hat{\theta}$  is given by

$$\mathcal{I}_{\text{up,param}}(z) \stackrel{\text{def}}{=} \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} \Big|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta}), \quad (1)$$

where  $H_{\hat{\theta}} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \nabla_{\theta}^2 L(z_i, \hat{\theta})$  is the Hessian and is positive definite (PD) by assumption. In essence, we are forming a quadratic approximation to the empirical risk around  $\hat{\theta}$  and take a single Newton step; see appendix A for a derivation. Since removing a point  $z$  is the same as upweighting it by  $\epsilon = -\frac{1}{n}$ , we can linearly approximate the parameter change due to removing  $z$  without retraining the model by computing  $\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n} \mathcal{I}_{\text{up,param}}(z)$ .

Next, we apply the chain rule to measure how upweighting  $z$  changes functions of  $\hat{\theta}$ . In particular, the influence of upweighting  $z$  on the loss at a test point  $z_{\text{test}}$  again has a closed-form expression:

$$\begin{aligned} \mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) &\stackrel{\text{def}}{=} \frac{dL(z_{\text{test}}, \hat{\theta}_{\epsilon,z})}{d\epsilon} \Big|_{\epsilon=0} \\ &= \nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^T \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} \Big|_{\epsilon=0} \\ &= -\nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta}). \end{aligned} \quad (2)$$

<sup>1</sup>We fold in any regularization terms into  $L$ .

### 2.2. Perturbing a training input

Let us develop a finer-grained notion of influence by studying a different counterfactual: how would the model's predictions change if a training input were modified?

For a training point  $z = (x, y)$ , define  $z_{\delta} \stackrel{\text{def}}{=} (x + \delta, y)$ . Consider the perturbation  $z \mapsto z_{\delta}$ , and let  $\hat{\theta}_{z_{\delta}, -z}$  be the empirical risk minimizer on the training points with  $z_{\delta}$  in place of  $z$ . To approximate its effects, define the parameters resulting from moving  $\epsilon$  mass from  $z$  onto  $z_{\delta}$ :  $\hat{\theta}_{\epsilon,z_{\delta}, -z} \stackrel{\text{def}}{=} \arg \min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n L(z_i, \theta) + \epsilon L(z_{\delta}, \theta) - \epsilon L(z, \theta)$ . An analogous calculation to (1) yields:

$$\begin{aligned} \frac{d\hat{\theta}_{\epsilon,z_{\delta}, -z}}{d\epsilon} \Big|_{\epsilon=0} &= \mathcal{I}_{\text{up,param}}(z_{\delta}) - \mathcal{I}_{\text{up,param}}(z) \\ &= -H_{\hat{\theta}}^{-1} (\nabla_{\theta} L(z_{\delta}, \hat{\theta}) - \nabla_{\theta} L(z, \hat{\theta})). \end{aligned} \quad (3)$$

As before, we can make the linear approximation  $\hat{\theta}_{z_{\delta}, -z} - \hat{\theta} \approx \frac{1}{n} (\mathcal{I}_{\text{up,param}}(z_{\delta}) - \mathcal{I}_{\text{up,param}}(z))$ , giving us a closed-form estimate of the effect of  $z \mapsto z_{\delta}$  on the model. Analogous equations also apply for changes in  $y$ . While influence functions might appear to only work for infinitesimal (therefore continuous) perturbations, it is important to note that this approximation holds for arbitrary  $\delta$ : the  $\epsilon$ -upweighting scheme allows us to smoothly interpolate between  $z$  and  $z_{\delta}$ . This is particularly useful for working with discrete data (e.g., in NLP) or with discrete label changes.

If  $x$  is continuous and  $\delta$  is small, we can further approximate (3). Assume that the input domain  $\mathcal{X} \subseteq \mathbb{R}^d$ , the parameters  $\Theta \subseteq \mathbb{R}^p$ , and  $L$  is differentiable in  $\theta$  and  $x$ . As  $\|\delta\| \rightarrow 0$ ,  $\nabla_{\theta} L(z_{\delta}, \hat{\theta}) - \nabla_{\theta} L(z, \hat{\theta}) \approx [\nabla_x \nabla_{\theta} L(z, \hat{\theta})]\delta$ , where  $\nabla_x \nabla_{\theta} L(z, \hat{\theta}) \in \mathbb{R}^{p \times d}$ . Substituting into (3),

$$\frac{d\hat{\theta}_{\epsilon,z_{\delta}, -z}}{d\epsilon} \Big|_{\epsilon=0} \approx -H_{\hat{\theta}}^{-1} [\nabla_x \nabla_{\theta} L(z, \hat{\theta})]\delta. \quad (4)$$

We thus have  $\hat{\theta}_{z_{\delta}, -z} - \hat{\theta} \approx -\frac{1}{n} H_{\hat{\theta}}^{-1} [\nabla_x \nabla_{\theta} L(z, \hat{\theta})]\delta$ . Differentiating w.r.t.  $\delta$  and applying the chain rule gives us

$$\begin{aligned} \mathcal{I}_{\text{pert,loss}}(z, z_{\text{test}}) &\stackrel{\text{def}}{=} \nabla_{\theta} L(z_{\text{test}}, \hat{\theta})_{z_{\delta}, -z} \Big|_{\delta=0} \\ &= -\nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_x \nabla_{\theta} L(z, \hat{\theta}). \end{aligned} \quad (5)$$

$[\mathcal{I}_{\text{pert,loss}}(z, z_{\text{test}})]\delta$  tells us the approximate effect that  $z \mapsto z + \delta$  has on the loss at  $z_{\text{test}}$ . By setting  $\delta$  in the direction of  $\mathcal{I}_{\text{pert,loss}}(z, z_{\text{test}})^T$ , we can construct local perturbations of  $z$  that maximally increase the loss at  $z_{\text{test}}$ . In Section 5.2, we will use this to construct training-set attacks. Finally, we note that  $\mathcal{I}_{\text{pert,loss}}(z, z_{\text{test}})$  can help us identify the features of  $z$  that are most responsible for the prediction on  $z_{\text{test}}$ .

### 2.3. Relation to Euclidean distance

To find the training points most relevant to a test point, it is common to look at its nearest neighbors in Euclidean

(ب) چالش ها و راه حل ها (بخش ۳):

جزئیات کامل چگونگی محاسبات در صفحه سوم و چهارم مقاله آمده است. در اینجا به اختصار به توضیح آن می پردازیم. سپس قسمت های مهم مقاله را نیز نشان میدهیم.

چالش ها:

- هزینه محاسباتی (Computational Cost): معکوس کردن مستقیم  $H_{\hat{\theta}}$  به عملیات  $O(p^3)$  نیاز دارد (که در آن  $p$  تعداد پارامترها است)، که برای مدل های بزرگ منعو و غیرممکن است.
- مقیاس شدن به مجموعه داده های بزرگ (Scaling to Large Datasets): محاسبه influence برای همه نقاط آموختی گران است زیرا شامل محاسبه گرادیان ها و Hessian-vector products است.

راه حل:

به جای اینکه به طور صریح HVPs Implicit Hessian-Vector Products را تشکیل یا معکوس کنیم، از HVP ها برای محاسبه های مانند  $H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z_{\text{test}}, \hat{\theta})$  به طور موثر در زمان  $O(p)$  در هر تکرار استفاده می شود.

روش گرادیان مزدوج (Conjugate Gradient (CG) Method): این روش مسئله معکوس ماتریس را به یک تسلیک بهینه سازی تبدیل می کند و  $H_{\hat{\theta}} v = \nabla_{\theta} L(z, \hat{\theta})$  را بصورت مکرر حل می کند.

تخمین تصادفی (Stochastic Estimation): برای سرعت بخشیدن به CG برای مجموعه داده های بزرگ، تنها زیر مجموعه ای از نقاط آموزشی در هر تکرار نمونه برداری می شود، که  $H_{\hat{\theta}}$  را با برآوردگرهای unbiased تقریب می زند.

این روش ها پیچیدگی محاسباتی را به  $O(np + rtp)$  کاهش می دهند، جایی که  $n$  اندازه مجموعه داده،  $p$  تعداد پارامترها،  $t$  تعداد تکرارهای CG و  $r$  تعداد تکرارها است. قسمت های مقاله از مقاله که به تفسیر این موضوع می پردازند در ادامه آمدند:

is too expensive for models like deep neural networks with millions of parameters. Second, we need to calculate  $\mathcal{I}_{\text{up,loss}}(z_i, z_{\text{test}})$  across all training points  $z_i$ .

The first problem is well-studied in second-order optimization. The idea is to avoid explicitly computing  $H_{\hat{\theta}}^{-1}$ ; instead, we use implicit Hessian-vector products (HVPs) to efficiently approximate  $s_{\text{test}} \stackrel{\text{def}}{=} H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z_{\text{test}}, \hat{\theta})$  and then compute  $\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -s_{\text{test}} \cdot \nabla_{\theta} L(z, \hat{\theta})$ . This also solves the second problem: for each test point of interest, we can precompute  $s_{\text{test}}$  and then efficiently compute  $-s_{\text{test}} \cdot \nabla_{\theta} L(z_i, \hat{\theta})$  for each training point  $z_i$ .

We discuss two techniques for approximating  $s_{\text{test}}$ , both relying on the fact that the HVP of a single term in  $H_{\hat{\theta}}$ ,  $[\nabla_{\theta}^2 L(z_i, \hat{\theta})]v$ , can be computed for arbitrary  $v$  in the same time that  $\nabla_{\theta} L(z_i, \hat{\theta})$  would take, which is typically  $O(p)$  (Pearlmutter, 1994).

**Conjugate gradients (CG).** The first technique is a standard transformation of matrix inversion into an optimization problem. Since  $H_{\hat{\theta}} \succ 0$  by assumption,  $H_{\hat{\theta}}^{-1} v \equiv \arg \min_t \{t^T H_{\hat{\theta}} t - v^T t\}$ . We can solve this with CG approaches that only require the evaluation of  $H_{\hat{\theta}} t$ , which takes  $O(np)$  time, without explicitly forming  $H_{\hat{\theta}}$ . While an exact solution takes  $p$  CG iterations, in practice we can get a good approximation with fewer iterations; see Martens (2010) for more details.

**Stochastic estimation.** With large datasets, standard CG can be slow; each iteration still goes through all  $n$  training points. We use a method developed by Agarwal et al. (2016) to get an estimator that only samples a single point per iteration, which results in significant speedups.

### 3. Efficiently calculating influence

There are two challenges to efficiently computing  $\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta})$ . First, it requires forming and inverting  $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^n \nabla_{\theta}^2 L(z_i, \hat{\theta})$ , the Hessian of the empirical risk. With  $n$  training points and  $\theta \in \mathbb{R}^p$ , this requires  $O(np^2 + p^3)$  operations, which

Dropping the  $\hat{\theta}$  subscript for clarity, let  $H_j^{-1} \stackrel{\text{def}}{=} \sum_{i=0}^j (I - H)^i$ , the first  $j$  terms in the Taylor expansion of  $H^{-1}$ . Rewrite this recursively as  $H_j^{-1} = I + (I - H)H_{j-1}^{-1}$ . From the validity of the Taylor expansion,  $H_j^{-1} \rightarrow H^{-1}$  as  $j \rightarrow \infty$ .<sup>2</sup> The key is that at each iteration, we can substitute the full  $H$  with a draw from any unbiased (and faster-to-compute) estimator of  $H$  to form  $\tilde{H}_j$ . Since  $\mathbb{E}[\tilde{H}_j^{-1}] = H_j^{-1}$ , we still have  $\mathbb{E}[\tilde{H}_j^{-1}] \rightarrow H^{-1}$ .

In particular, we can use  $\nabla_\theta^2 L(z_i, \hat{\theta})$ , for any  $z_i$ , as an unbiased estimator of  $H$ . This gives us the following procedure: uniformly sample  $t$  points  $z_{s_1}, \dots, z_{s_t}$  from the training data; define  $\tilde{H}_0^{-1}v = v$ ; and recursively compute  $\tilde{H}_j^{-1}v = v + (I - \nabla_\theta^2 L(z_{s_j}, \hat{\theta}))\tilde{H}_{j-1}^{-1}v$ , taking  $\tilde{H}_t^{-1}v$  as our final unbiased estimate of  $H^{-1}v$ . We pick  $t$  to be large enough such that  $\tilde{H}_t$  stabilizes, and to reduce variance we repeat this procedure  $r$  times and average results. Empirically, we found this significantly faster than CG.

We note that the original method of Agarwal et al. (2016) dealt only with generalized linear models, for which  $[\nabla_\theta^2 L(z_i, \hat{\theta})]v$  can be efficiently computed in  $O(p)$  time. In our case, we rely on Pearlmutter (1994)'s more general algorithm for fast HVPs, described above, to achieve the same time complexity.<sup>3</sup>

With these techniques, we can compute  $I_{\text{up,loss}}(z_i, z_{\text{test}})$  on all training points  $z_i$  in  $O(np + rtp)$  time; we show in Section 4.1 that empirically, choosing  $rt = O(n)$  gives accurate results. Similarly, we can compute  $I_{\text{per,loss}}(z_i, z_{\text{test}}) = -\frac{1}{n} \nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_x \nabla_\theta L(z_i, \hat{\theta})$  with two matrix-vector products: we first compute  $s_{\text{test}}$ , then find  $s_{\text{test}}^\top \nabla_x \nabla_\theta L(z_i, \hat{\theta})$  with the same HVP trick. These computations are easy to implement in auto-grad systems like TensorFlow (Abadi et al., 2015) and Theano (Theano D. Team, 2016), as users need only specify the loss; the rest is automatically handled.

(ج) معنی دار بودن تابع influence در مدل های غیر محدب و دو بار مشتق پذیر نبودن تابع هزینه (بخش ۴):

جزیيات کامل چگونگی محاسبات در صفحه چهارم و پنجم مقاله آمده است. در اینجا به اختصار به توضیح آن می پردازیم. سپس قسمت های مهم مقاله را نیز نشان میدهیم.

در حالی که مشتقهای نظری (theoretical derivations) توابع influence (convexity) و twice-convexity (twice-convexity) را فرض می کنند، تقریب ها (approximations) حق در عمل برای شبکه های عصبی غیر محدب مفید باقی میمانند. (در صفحه پنج مقاله یک مدل را تست کرده اند و نتایج مفید بودن approximation ها را support میکنند).

برای non-convexity، یک Hessian  $(H_{\hat{\theta}} + \lambda I)$  اضافه می شود تا اثبات عددی اطمینان حاصل شود. این به توابع تأثیر اجازه می دهد تا نتایج معنی داری ارائه دهنده حقیقت زمانی که  $\hat{\theta}$  فقط یک حداقل محلی باشد.

#### 4.2. Non-convexity and non-convergence

In Section 2, we took  $\hat{\theta}$  as the global minimum. In practice, if we obtain our parameters  $\hat{\theta}$  by running SGD with early stopping or on non-convex objectives,  $\hat{\theta} \neq \theta$ . As a result,  $H_{\hat{\theta}}$  could have negative eigenvalues. We show that influence functions on  $\hat{\theta}$  still give meaningful results in practice.

Our approach is to form a convex quadratic approximation of the loss around  $\hat{\theta}$ , i.e.,  $\tilde{L}(z, \theta) = L(z, \hat{\theta}) + \nabla L(z, \hat{\theta})^\top (\theta - \hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^\top (H_{\hat{\theta}} + \lambda I)(\theta - \hat{\theta})$ . Here,  $\lambda$  is a damping term that we add if  $H_{\hat{\theta}}$  has negative eigenvalues;

برای influence (به عنوان مثال، smooth approximations، non-differentiable losses) برای محاسبه توابع influence استفاده می شود. این تقریب ها اغلب با واقعی influence مطابقت دارند، همانطور که در آزمایش ها نشان داده شده است.

#### 4.3. Non-differentiable losses

What happens when the derivatives of the loss,  $\nabla_{\theta} L$  and  $\nabla_{\theta}^2 L$ , do not exist? In this section, we show that influence functions computed on smooth approximations to non-differentiable losses can predict the behavior of the original, non-differentiable loss under leave-one-out retraining. The robustness of this approximation suggests that we can train non-differentiable models and swap out non-differentiable components for smoothed versions for the purposes of calculating influence.

To see this, we trained a linear SVM on the same 1s vs. 7s MNIST task in Section 2.3. This involves minimizing  $\text{Hinge}(s) = \max(0, 1 - s)$ ; this simple piece-

استحکام influence function نسبت به این مفروضات، آن را برای کاربردهای شبکه عصبی در دنیای واقعی عملی می کند. قسمت های مقاله از مقاله که به تفسیر این موضوع می پردازند در ادامه آمده اند:

provide useful information even when these assumptions are violated (Sections 4.2, 4.3).

#### 4.1. Influence functions vs. leave-one-out retraining

Influence functions assume that the weight on a training point is changed by an infinitesimally small  $\epsilon$ . To investigate the accuracy of using influence functions to approximate the effect of removing a training point and retraining, we compared  $-\frac{1}{n} \mathcal{I}_{\text{up}, \text{loss}}(z_i, z_{\text{test}})$  with  $L(z_{\text{test}}, \hat{\theta}_{-i}) - L(z_{\text{test}}, \hat{\theta})$  (i.e., actually doing leave-one-out retraining). With a logistic regression model on 10-class MNIST,<sup>4</sup> the predicted and actual changes matched closely (Fig 2-Left).

The stochastic approximation from Agarwal et al. (2016) was also accurate with  $r = 10$  repeats and  $t = 5,000$  iterations (Fig 2-Mid). Since each iteration only requires one HVP  $[\nabla_{\theta}^2 L(z_i, \hat{\theta})]v$ , this runs quickly: in fact, we accurately estimated  $H^{-1}v$  without even looking at every data point, since  $n = 55,000 > rt$ . Surprisingly, even  $r = 1$  worked; while results were noisier, it was still able to identify the most influential points.

#### 4.2. Non-convexity and non-convergence

In Section 2, we took  $\hat{\theta}$  as the global minimum. In practice, if we obtain our parameters  $\tilde{\theta}$  by running SGD with early stopping or on non-convex objectives,  $\tilde{\theta} \neq \hat{\theta}$ . As a result,  $H_{\tilde{\theta}}$  could have negative eigenvalues. We show that influence functions on  $\tilde{\theta}$  still give meaningful results in practice.

Our approach is to form a convex quadratic approximation of the loss around  $\tilde{\theta}$ , i.e.,  $\tilde{L}(z, \theta) = L(z, \tilde{\theta}) + \nabla L(z, \tilde{\theta})^T(\theta - \tilde{\theta}) + \frac{1}{2}(\theta - \tilde{\theta})^T(H_{\tilde{\theta}} + \lambda I)(\theta - \tilde{\theta})$ . Here,  $\lambda$  is a damping term that we add if  $H_{\tilde{\theta}}$  has negative eigenvalues;

## 4. Validation and extensions

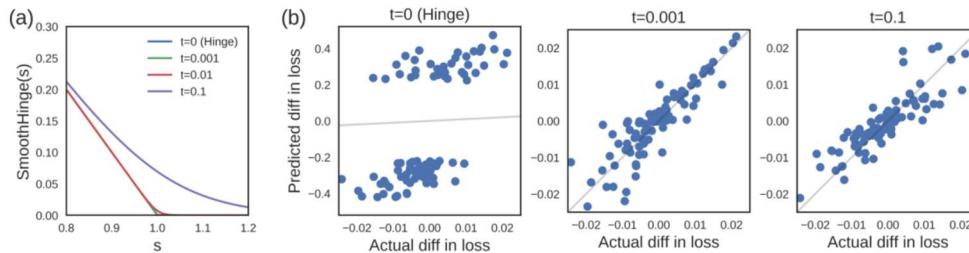
Recall that influence functions are asymptotic approximations of leave-one-out retraining under the assumptions that (i) the model parameters  $\hat{\theta}$  minimize the empirical risk, and that (ii) the empirical risk is twice-differentiable and strictly convex. Here, we empirically show that influence functions are accurate approximations (Section 4.1) that

<sup>2</sup>We assume w.l.o.g. that  $\forall i, \nabla_{\theta}^2 L(z_i, \hat{\theta}) \preccurlyeq I$ ; if this is not true, we can scale the loss down without affecting the parameters. In some cases, we can get an upper bound on  $\nabla_{\theta}^2 L(z_i, \hat{\theta})$  (e.g., for linear models and bounded input), which makes this easy. Otherwise, we treat the scaling as a separate hyperparameter and tune it such that the Taylor expansion converges.

<sup>3</sup>To increase stability, especially with non-convex models (see Section 4.2), we can also sample a minibatch of training points at each iteration, instead of relying on a single training point.

wise linear function is similar to ReLUs, which cause non-differentiability in neural networks. We set the derivatives at the hinge to 0 and calculated  $\mathcal{I}_{\text{up,loss}}$ . As one might expect, this was inaccurate (Fig 3b-Left): the second derivative carries no information about how close a support vector  $z$  is to the hinge, so the quadratic approximation of  $L(z, \hat{\theta})$  is linear, which leads to  $\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}})$  overestimating the influence of  $z$ .

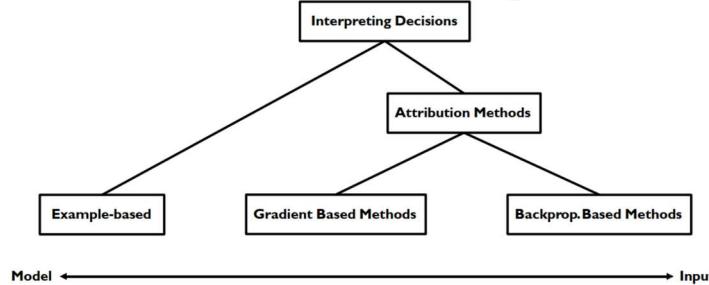
For the purposes of calculating influence, we approximated Hinge( $s$ ) with SmoothHinge( $s, t$ ) =  $t \log(1 + \exp(\frac{1-s}{t}))$ , which approaches the hinge loss as  $t \rightarrow 0$  (Fig 3a). Using the same SVM weights as before, we found that calculating  $\mathcal{I}_{\text{up,loss}}$  using SmoothHinge( $s, 0.001$ ) closely matched the actual change due to retraining in the original Hinge( $s$ ) (Pearson's R = 0.95; Fig 3b-Mid) and remained accurate over a wide range of  $t$  (Fig 3b-Right).



**Figure 3. Smooth approximations to the hinge loss.** (a) By varying  $t$ , we can approximate the hinge loss with arbitrary accuracy: the green and blue lines are overlaid on top of each other. (b) Using a random, wrongly-classified test point, we compared the predicted vs. actual differences in loss after leave-one-out retraining on the 100 most influential training points. A similar trend held for other test points. The SVM objective is to minimize  $0.005 \|w\|_2^2 + \frac{1}{n} \sum_i \text{Hinge}(y_i w^\top x_i)$ . **Left:** Influence functions were unable to accurately predict the change, overestimating its magnitude considerably. **Mid:** Using SmoothHinge( $\cdot, 0.001$ ) let us accurately predict the change in the hinge loss after retraining. **Right:** Correlation remained high over a wide range of  $t$ , though it degrades when  $t$  is too large. When  $t = 0.001$ , Pearson's R = 0.95; when  $t = 0.1$ , Pearson's R = 0.91.

## Attribution Methods ↴

### Types of DNN Interpretability



(آ) معیار اندازه‌گیری تأثیر هر مولفه ورودی

معیار:

تأثیر هر مولفه ورودی در روش‌های attribution با گرادیان خروجی مدل نسبت به ورودی اندازه‌گیری می‌شود. این معیار میزان حساسیت خروجی به تغییرات کوچک در هر ویژگی ورودی را تعیین می‌کند.

- روش‌های Attribution، تأثیر مؤلفه‌های ورودی را با استفاده از سهم آنها در خروجی مدل اندازه‌گیری می‌کنند.
- این معمولاً از طریق مشتق‌ات جزئی (gradients) خروجی با توجه به هر ویژگی ورودی اندازه‌گیری می‌شود.

$$\text{Attribution for } x_i = \frac{\partial f(x)}{\partial x_i}$$

Example-based

Attribution Methods

Gradient Based

Backprop. Based

#### The Baseline Attribution Method Saliency Map

- Gradient of the decision  $f(x)$  with respect to each input image  $x$ :

$$\text{Saliency}(x) := \nabla_x f(x) = \frac{\partial f(x)}{\partial x}$$

- Can be calculated through backpropagation.

منطقی بودن:

گرادیان‌ها یک انتخاب طبیعی معقول هستند زیرا مستقیماً نزدیک تغییر local در خروجی ناشی از اختلالات ورودی را اندازه‌گیری می‌کنند (directly measure the local rate of change in the output caused by perturbations to the input).

ویژگی‌هایی با اندازه گرادیان بزرگ‌تر تأثیرگذارتر تلقی می‌شوند، زیرا تأثیر قوی‌تری بر پیش‌بینی دارند. در واقع، گرادیان ورودی حساسیت خروجی مدل را نسبت به تغییرات در ورودی اندازه‌گیری می‌کند و مشخص می‌کند که کدام ویژگی بیشترین کمک را به پیش‌بینی دارد.

- گرادیان ورودی مستقیماً نحوه تأثیر تغییرات ورودی بر خروجی را نشان می‌دهد.
- با درک شهودی ما از اهمیت مطابقت دارد.
- از نظر ریاضی قابل انجام است و می‌توان آن را به طور موثر محاسبه کرد.

روش محاسبه در شبکه‌های عصبی:

گرادیان‌ها از طریق backpropagation محاسبه می‌شوند، که به طور موثر مشتق‌ات جزئی را از طریق شبکه با استفاده از قانون زنجیری مشتق (chain rule) محاسبه می‌کند. چارچوب‌های یادگیری عمیق مدرن (مانند PyTorch، TensorFlow) این فرآیند را خودکار می‌کنند. همچنین در نظر داریم که روش‌های پیچیده‌تری هم موجود اند که در طول فرایند backpropagation کارهای vanilla attribution و Deconvolution Guided Backprop انجام می‌کنند تا نتایج بهتر شوند مانند gradient backpropagation است که با استفاده از gradient backpropagation محاسبه می‌شود:

- خروجی را محاسبه می‌کنیم: Forward pass
- $\frac{\partial y}{\partial x_i}$  را برای هر ورودی  $x$  محاسبه می‌کنیم: Backward pass

مثال عددی:

یک شبکه عصبی کوچک با یک نورون را در نظر بگیرید:

:ForwardPass

- Function:  $f(x_1, x_2) = \text{sigmoid}(2x_1 - x_2 + 1)$
- Input:  $x_1 = 1, x_2 = 2$
- Pre-activation:  $2(1) - 1(2) + 1 = 1$
- Output:  $\text{sigmoid}(1) \approx 0.731$

(محاسبه گرادیان نسبت به ورودی): Backpropagation

- Derivative of sigmoid at  $z = 1$ :  $\sigma'(1) = \sigma(1)(1 - \sigma(1)) \approx 0.731 \times 0.269 \approx 0.197$
- Gradient w.r.t  $x_1$ :  $0.197 \times 2 = 0.394$
- Gradient w.r.t  $x_2$ :  $0.197 \times (-1) = -0.197$

تفسیر:

$x_1$  دارای یک attribution مثبت است (خروجی را افزایش می دهد)، در حالی که  $x_2$  یک attribution منفی دارد (خروجی را کاهش می دهد). همچنین قدر مطلق اندازه گرادیان  $x_1$  بیشتر از  $x_2$  است پس در محاسبه خروجی  $x_1$  دارای contribution بیشتری است.

(ب)

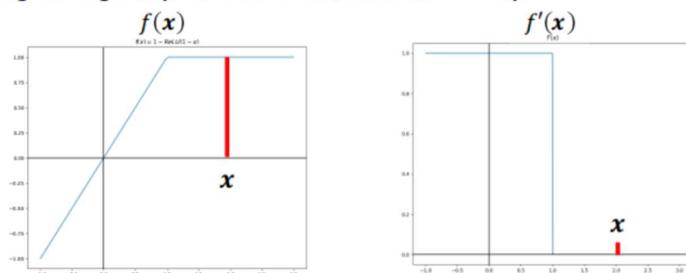
Example-based	Attribution Methods	Gradient Based	Backprop. Based
<b>Summary</b>	<ul style="list-style-type: none"> <li>- Attribution method assigns “attribution score” to each input pixel.</li> <li>- Baseline attribution method Saliency Map is noisy.</li> <li>- Hypothesis 1: Saliency maps are truthful.</li> <li>- Hypothesis 2: Gradients are discontinuous.</li> <li>- Hypothesis 3: <math>f(x)</math> saturates.</li> <li>- Two solution approaches: Gradient-based method and Backprop-based method.</li> </ul>		

. مشکل ۱. : Gradient Saturation

- چالش: در saturated regions یا مناطق اشباع شده (به عنوان مثال، مناطق مسطح توایع فعال سازی مانند سیگموئید یا ReLU)، گرادیان ها به صفر نزدیک می شوند، که به اشتباہ به معنای عدم تأثیر ورودی است.

Hypothesis 3 –  $f(x)$  saturates

- A feature may have a strong effect globally, but with a small derivative locally.



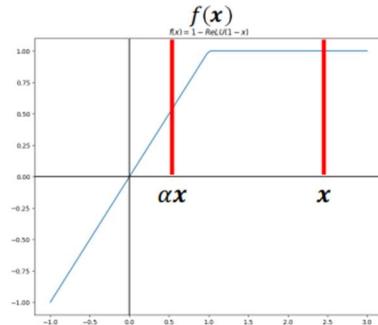
راه حل:

روش اول: از روش Interior Gradient که با استفاده از فرمول زیر توابع را از قسمت های مسطح که گرادیان صفر هستند خارج میکند استفاده کنیم.

## 2. Interior Gradient Hypothesis 3 – $f(x)$ saturates

$$IntGrad(\mathbf{x}) := \frac{\partial f(\mathbf{x}^*)}{\partial \mathbf{x}^*}, \quad \mathbf{x}^* = \alpha \mathbf{x}, \quad 0 < \alpha \leq 1$$

-Appropriate  $\alpha$  will trigger informative activation functions



روش دوم: از Integrated Gradients استفاده کنیم، که average gradient را در طول یک مسیر از یک baseline (به عنوان مثال، zero input) تا ورودی واقعی محاسبه می کند:

$$IG(x) = (x - x_{\text{baseline}}) \int_0^1 \frac{\partial f(x_{\text{baseline}} + \alpha(x - x_{\text{baseline}}))}{\partial x} d\alpha$$

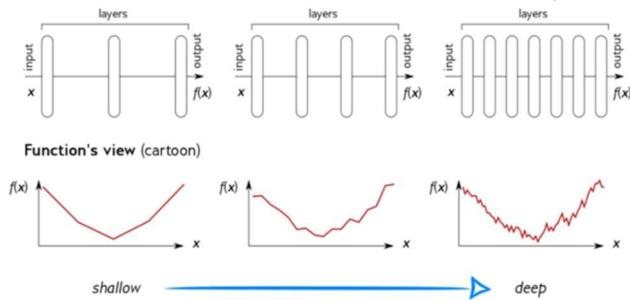
مشکل ۲. Gradients are discontinuous.

چالش: در شبکه عصبی های عمیق به علت استفاده از piecewise-linear functions همچون ReLU activation, max-pooling, etc تابع گرادیان ناپیوستگی دارد که منجر به جهش ناگهانی در امتیاز اهمیت نسبت به تغییرات بی نهایت کوچک در ورودی می شود.

## Hypothesis 2 – Gradients are discontinuous

- DNN uses piecewise-linear functions (ReLU activation, max-pooling, etc.).
- Sudden jumps in the importance score over infinitesimal changes in the input.

بی نهایت کوچک

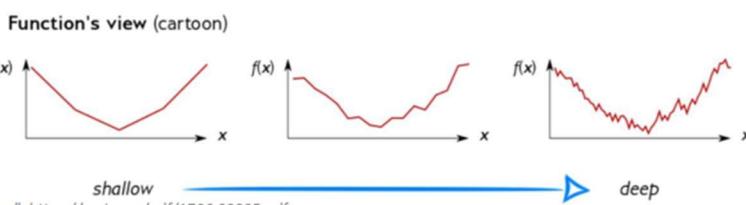


راه حل: از روش SmoothGrad که با استفاده از فرمول زیر smooth gradient را میکند استفاده کنیم. اساس این کار اضافه کردن نویز به ورودی است. (SmoothGrad averages gradients over multiple noisy perturbations of the input)

## I. SmoothGrad Hypothesis 2 – Gradients are discontinuous: Just smooth the gradient!

$$SmoothGrad(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^n \frac{\partial f(\mathbf{x}^*)}{\partial \mathbf{x}^*}, \quad \mathbf{x}^* = \mathbf{x} + \mathcal{N}(0, \sigma^2)$$

میانگین Gaussian smoothing



مشکل ۳. (Noisy/Unreliable/Instable Gradients) Gradients are very Noisy!  
 چالش: گرادیان ها می توانند به دلیل ورودی های با ابعاد بالا یا توابع non-smooth، بسیار noisy باشند که منجر به inconsistent attribution می شود.

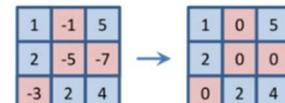
راه حل: با استفاده از روش های Backpropagation که در هنگام Backprop-based کنند مانند Guided Backpropagation یا Deconvnet بعضی از گرادیان ها را حذف می کنیم. حذف گرادیان های بیشتر منجر به visualization واضح تر می شود که دارای نویز کمتری است. نحوه کار این دو روش در ادامه آمده است.

pass the reconstructed signal through ReLUs :Deconvnet

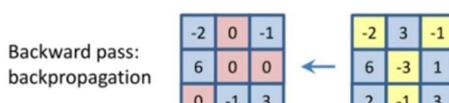
Removing negative gradient + consider forward activations :Guided Backpropagation

### I. Deconvnet

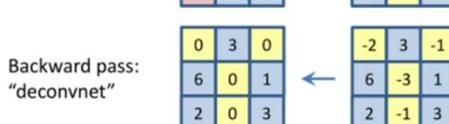
- Maps feature pattern to input space (image reconstruction)



- To obtain valid feature reconstruction, pass the reconstructed signal through ReLUs



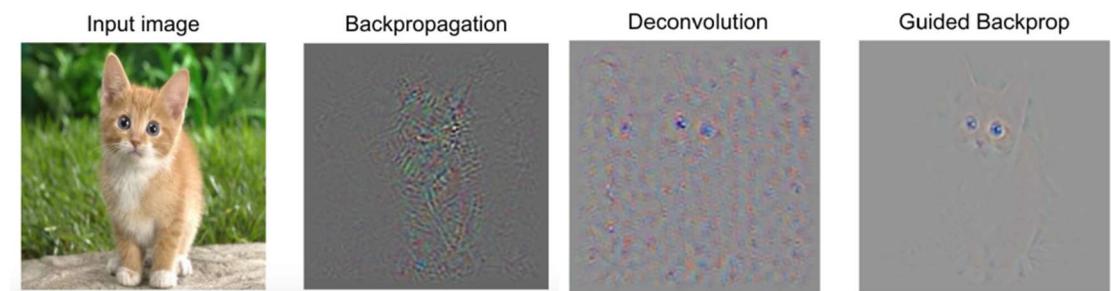
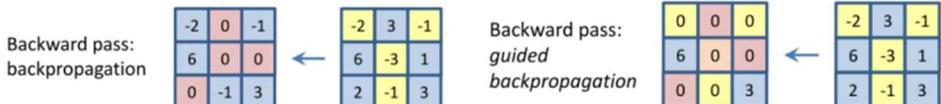
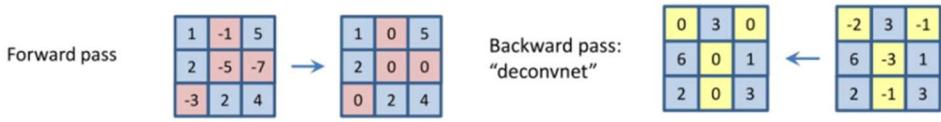
- Removing noise by removing negative gradient



### 2. Guided Backpropagation

- Combine Deconvnet with Backpropagation

- Removing negative gradient + consider forward activations



**Observation:** Removing more gradient leads to sharper visualizations

### 4. Local vs Global Explanations

چالش: گرادیان های ساده فقط توضیحات محلی ارائه می دهند که ممکن است اهمیت ویژگی global را دربر نگیرند.

راه حل: Layer-wise Relevance Propagation (LRP) را از طریق شبکه به عقب منتشر می کند.

relevance scores ○  
 ارتباط کلی بین لایه ها را حفظ می کند.

با در نظر گرفتن کل ساختار شبکه global explanation های بیشتری ارائه می دهد.

##### ۵. مشکل Computational Complexity :

- چالش: محاسبه گرادیان یا سایر معیارهای attribution برای ورودی های با ابعاد بالا مانند تصاویر می تواند از نظر محاسباتی گران باشد.

- راه حل: از تکنیک های تقریبی مانند Grad-CAM استفاده کنیم، که گرادیان ها را در لایه های بالاتر شبکه aggregate می کند و سریار محاسباتی را کاهش می دهد و در عین حال قابلیت تفسیرپذیری را حفظ می کند.

هر یک از این راه حل ها به محدودیت خاصی از روش های basic gradient-based attribution می پردازد و در عین حال کارایی محاسباتی و درستی ریاضی را حفظ می کند. انتخاب راه حلی که برای استفاده از آن استفاده شود اغلب به کاربرد و الزامات خاص کار تفسیر بستگی دارد.

### سوال سوم

.1

(۱)

Attention Gates (AGs) در معماری U-Net به طور تطبیقی بر روی مناطق تصویر مربوطه تمرکز می‌کنند و توانایی مدل را برای segmentation اندام‌ها با تغییرات قابل توجه در شکل و اندازه بهبود می‌بخشند. به عبارت دیگر، Attention Gates یا دروازه‌های توجه (AGs) توانایی U-Net را برای مدیریت تغییرات در شکل و اندازه اندام با تمرکز انتخابی (selectively focusing) بر روی مناطق تصویر مرتبط و مهم و در عین حال سرکوب مناطق نامریوط افزایش می‌دهند.

به طور مشخص اثرات آن شامل موارد زیر است:

- انتخاب ویژگی هدفمند (Targeted Feature Selection): AG‌ها ویژگی‌های پس‌زمینه نامریوط را فیلتر می‌کنند و بر موارد مربوطه تأکید می‌کنند و اطمینان حاصل می‌کنند که شبکه بر ساختارهای مورد علاقه تمرکز و توجه می‌کند، صرف نظر از تغییر اندازه و شکل آنها. آنها با یادگیری ضرایب توجه متفاوت مکانی به این امر دست می‌یابند، که فعال سازی ویژگی‌ها را بر اساس اهمیت آنها برای وظیفه تقسیم بندی تعديل می‌کند.
- آگاهی زمینه‌ای (Contextual Awareness): با استفاده از اطلاعات زمینه‌ای از مقیاس‌های درشت‌تر، AG شبکه را قادر می‌سازد تا بازنمایی ویژگی‌ها را برای segmentation بهتر اندام‌ها که در بیماران مختلف متفاوت است، اصلاح کند.
- Dynamic Gating: AG‌ها ضرایب توجه را برای هر منطقه فضایی (spatial region) محاسبه می‌کنند و به صورت پویا با شکل‌ها و اندازه‌های اندام خاص در تصاویر ورودی تنظیم می‌شوند، که false positives را کاهش می‌دهد و دقت segmentation را افزایش می‌دهد. به عبارت دیگر، آنها با انطباق پویا با ویژگی‌های فضایی و زمینه‌ای ساختارهای هدف (مانند اندام‌ها) با استفاده از سیگنال‌های دروازه‌ای از لایه‌های درشت‌تر (gating signals from coarser layers) به این امر دست می‌یابند. این به مدل اجازه می‌دهد تا هم زمینه در مقیاس بزرگ و هم جزئیات دقیق را بهتر به تصویر بکشد، و نیاز به محلی‌سازی صریح منطقه مورد علاقه (explicit region-of-interest localization)، به ویژه برای اندام‌های بسیار متغیر مانند پانکراس مفید است.

معماری AGs در ادامه آمده است:

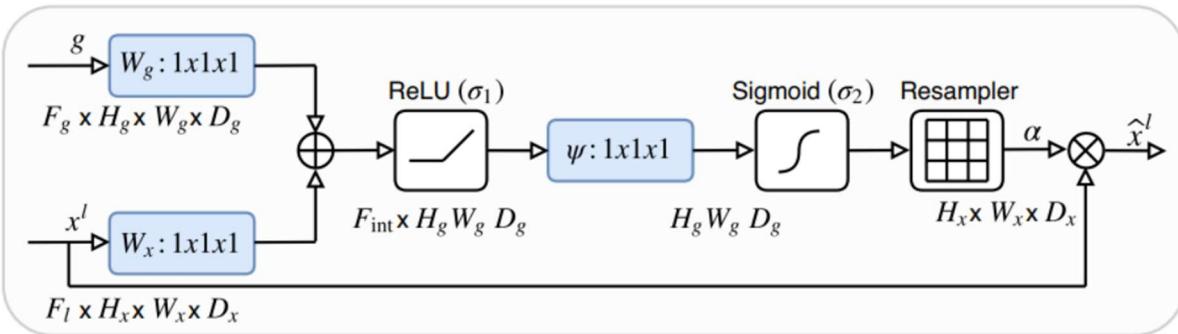


Figure 2: Schematic of the proposed additive attention gate (AG). Input features ( $x^l$ ) are scaled with attention coefficients ( $\alpha$ ) computed in AG. Spatial regions are selected by analysing both the activations and contextual information provided by the gating signal ( $g$ ) which is collected from a coarser scale. Grid resampling of attention coefficients is done using trilinear interpolation.

(ب) ابتدا بخشی از مقاله که این مکانیزم را معرفی کرده و فرمول آن را بیان کرده بینیم و سپس از روی آن توضیح دهیم:

Attention coefficients,  $\alpha_i \in [0, 1]$ , identify salient image regions and prune feature responses to preserve only the activations relevant to the specific task as shown in Figure 3a. The output of AGs is the element-wise multiplication of input feature-maps and attention coefficients:  $\hat{x}_{i,c}^l = x_{i,c}^l \cdot \alpha_i^l$ . In a default setting, a single scalar attention value is computed for each pixel vector  $x_i^l \in \mathbb{R}^{F_l}$  where  $F_l$  corresponds to the number of feature-maps in layer  $l$ . In case of multiple semantic classes, we propose to learn multi-dimensional attention coefficients. This is inspired by [29], where multi-dimensional attention coefficients are used to learn sentence embeddings. Thus, each AG learns to focus on a subset of target structures. As shown in Figure 2, a gating vector  $g_i \in \mathbb{R}^{F_g}$  is used for each pixel  $i$  to determine focus regions. The gating vector contains contextual information to prune lower-level feature responses as suggested in [32], which uses AGs for natural image classification. We use additive attention [2] to obtain the gating coefficient. Although this is computationally more expensive, it has experimentally shown to achieve higher accuracy than multiplicative attention [19]. Additive attention is formulated as follows:

$$q_{att}^l = \psi^T (\sigma_1 (W_x^T x_i^l + W_g^T g_i + b_g)) + b_\psi \quad (1)$$

$$\alpha_i^l = \sigma_2(q_{att}^l(x_i^l, g_i; \Theta_{att})), \quad (2)$$

where  $\sigma_2(x_{i,c}) = \frac{1}{1+exp(-x_{i,c})}$  correspond to sigmoid activation function. AG is characterised by a set of parameters  $\Theta_{att}$  containing: linear transformations  $W_x \in \mathbb{R}^{F_l \times F_{int}}$ ,  $W_g \in \mathbb{R}^{F_g \times F_{int}}$ ,  $\psi \in \mathbb{R}^{F_{int} \times 1}$  and bias terms  $b_\psi \in \mathbb{R}$ ,  $b_g \in \mathbb{R}^{F_{int}}$ . The linear transformations are computed using channel-wise 1x1x1 convolutions for the input tensors. In other contexts [33], this is referred to as **vector concatenation-based attention**, where the concatenated features  $x^l$  and  $g$  are linearly mapped to a  $\mathbb{R}^{F_{int}}$  dimensional intermediate space. In image captioning [1] and classification [11] tasks, the

softmax activation function is used to normalise the attention coefficients ( $\sigma_2$ ); however, sequential use of softmax yields sparser activations at the output. For this reason, we choose a sigmoid activation function. This results experimentally in better training convergence for the AG parameters. In contrast to [11] we propose a grid-attention technique. In this case, gating signal is not a global single vector for all image pixels but a grid signal conditioned to image spatial information. More importantly, the gating signal for each skip connection aggregates information from multiple imaging scales, as shown in Figure 1, which increases the grid-resolution of the query signal and achieve better performance. Lastly, we would like to note that AG parameters can be trained with the standard back-propagation updates without a need for sampling based update methods used in hard-attention [21].

**Attention Gates in U-Net Model:** The proposed AGs are incorporated into the standard U-Net architecture to highlight salient features that are passed through the skip connections, see Figure 1. Information extracted from coarse scale is used in gating to disambiguate irrelevant and noisy responses in skip connections. This is performed right before the concatenation operation to merge only relevant activations. Additionally, AGs filter the neuron activations during the forward pass as well as during the backward pass. Gradients originating from background regions are down weighted during the backward pass. This allows model parameters in shallower layers to be updated mostly based on spatial regions that are relevant to a given task. The update rule for convolution parameters in layer  $l - 1$  can be formulated as follows:

$$\frac{\partial(\hat{x}_i^l)}{\partial(\Phi^{l-1})} = \frac{\partial(\alpha_i^l f(x_i^{l-1}; \Phi^{l-1}))}{\partial(\Phi^{l-1})} = \alpha_i^l \frac{\partial(f(x_i^{l-1}; \Phi^{l-1}))}{\partial(\Phi^{l-1})} + \frac{\partial(\alpha_i^l)}{\partial(\Phi^{l-1})} x_i^l \quad (3)$$

The first gradient term on the right-hand side is scaled with  $\alpha_i^l$ . In case of multi-dimensional AGs,  $\alpha_i^l$  corresponds to a vector at each grid scale. In each sub-AG, complementary information is extracted and fused to define the output of skip connection. To reduce the number of trainable parameters

: فرمول بندی ریاضی

ضرایب دروازه (gating coefficients) ای  $\alpha_i$  را به صورت زیر محاسبه می کند:

۱. بازنمایی میانی (Intermediate Representation)

$$q_{att} = \Psi^T (\sigma(W_x^T x_i^l + W_g^T g_i + b_g)) + b_\psi$$

$x_i^l$ : ویژگی های ورودی در مکان فضایی  $i$  در لایه .i

$g_i$ : سیگنال دروازه ای (Gating signal) از مقیاس های درشت تر (coarser scales).

$W_x, W_g$ : به ترتیب وزن های تبدیل خطی برای ویژگی های ورودی و سیگنال های دروازه ای.

$b_g, b_\psi$ : عبارات بایاس.

$\sigma$ : تابع فعال سازی غیر خطی (ReLU).

## ۲. ضرایب توجه (Attention Coefficients)

$$\alpha_i = \sigma_2(q_{att})$$

$\sigma_2$ : تابع فعال‌سازی سیگموید برای نرمال‌سازی  $\alpha_i$  به  $[0, 1]$ .

## ۳. ویژگی‌های خروجی (Output Features)

$$x_i^l = x_i^l \cdot \alpha_i$$

$x_i^l$ : ویژگی‌های ورودی مقیاس‌بندی شده بر اساس ضرایب توجه (Scaled input features by attention) که در تووجه ضربی دیده می‌شود، جلوگیری می‌کند. همچنین، از مسائل محاسباتی مانند vanishing gradient های مرتبط با عملیات ضربی جلوگیری می‌کند.

(coefficients)

ترجیح توجه افزایشی بر توجه ضربی:

۱. استحکام در مقیاس (Robustness to Scale): توجه افزایشی از نظر محاسباتی پایدار (stable) است، به ویژه هنگامی که ویژگی‌های مقیاس یا بزرگ متفاوت را ترکیب می‌کند (combining features of different scales or magnitudes)، زیرا از exponential growth or decay issues که در تووجه ضربی دیده می‌شود، جلوگیری می‌کند. همچنین، از مسائل

محاسباتی مانند segmentation tasks های مرتبط با اندامها با شکل‌ها و اندازه‌های بسیار متغیر حیاتی است. این با همگرایی بهتر: از نظر تجربی، توجه افزایشی همگرایی آموزشی را برای AG ها بهبود می‌بخشد و آن را در تسک‌های پیش‌بینی متراکم (dense prediction tasks) مانند segmentation (dense prediction tasks) مؤثرتر می‌کند. به علاوه، در طول آموزش، به ویژه با داده

های تصویر پذشکی نویزی یا sparse، پایدارتر است.

۳. تمرکز مبتنی بر شبکه (Grid-Based Focus): توجه افزایشی امکان تمرکز localized focus بر روی مناطق (on regions) را فراهم می‌کند، که برای segmentation اندام‌ها با شکل‌ها و اندازه‌های بسیار متغیر حیاتی است. این با توجه sparse activations، که با multiplicative دست و پنجه نرم می‌کند، چالش‌برانگیزتر است.

(پ) مزایا:

۱. بهبود دقت Segmentation:

Attention U-Net ضرایب تشابه تاس (Dice Similarity Coefficients (DSC)) بالاتر و فاصله سطح به سطح (surface-to-surface distances (S2S)) کمتری را در مقایسه با روش‌های استاندارد U-Net و multi-modal می‌آورد.

به طور مداوم از U-Net استاندارد بهتر است، به ویژه برای ساختارهای کوچک و متغیر مانند پانکراس. به عنوان مثال، برای Segmentation DSC از ۰/۸۴۰ (U-Net) به ۰/۸۱۴ (Attention U-Net) بهبود یافته است.

با localizing بهتر مناطق هدف، recall را بهبود می‌بخشد. با سرکوب فعال‌سازی‌های پس‌زمینه نامربوط (irrelevant background activations)، موارد false positive را کاهش می‌دهد.

۲. کارایی یک مدل واحد (Single-Model Efficiency):

برخلاف مدل‌های آبشاری چند وجهی (multi-modal cascaded models) که به مراحل localization می‌آورند، Attention U-Net این فرآیندها را در یک مدل واحد ادغام می‌کند و باعث کاهش افروزنگی و سریار محاسباتی (reducing redundancy and computational overhead) می‌شود.

۳. Robustness در داده‌های آموزشی و داده‌های آموزشی محدود:

حتی با داده‌های آموزشی محدود نیز عملکرد خوبی دارد، به طور مداوم recall و precision بهتری را نسبت به U-Net استاندارد نشان می‌دهد، و آن را برای سناریوهایی با مجموعه داده‌های حاشیه نویسی کوچک مناسب می‌کند.

محدودیت ها:

#### ۱. هزینه های محاسباتی بالاتر:

- گنجاندن AG ها پارامترهای مدل را اندک (حدود ۸ درصد) افزایش می دهد و در مقایسه با استاندارد U-Net به منابع محاسباتی بیشتری برای آموختن نیاز دارد. (Requires additional memory and parameters)

#### ۲. وابستگی به وضوح ورودی (Dependency on Input Resolution):

- عملکرد مدل به وضوح تصویر حساس است. Downsampling ممکن است توانایی آن را برای در بر گرفتن جزئیات دقیق تر محدود کند، در حالی که چارچوب های چند وجهی (multi-modal frameworks) می توانند با استفاده از مارژول های جداگانه با وضوح بالا (using separate high-resolution modules) بر آن غلبه کنند.
- همچنین، Performance gain ممکن است برای تسك های ساده تر با اهداف well-contrasted (مانند کلیه ها) کاهش یابد.

#### ۳. طراحی Application-Specific

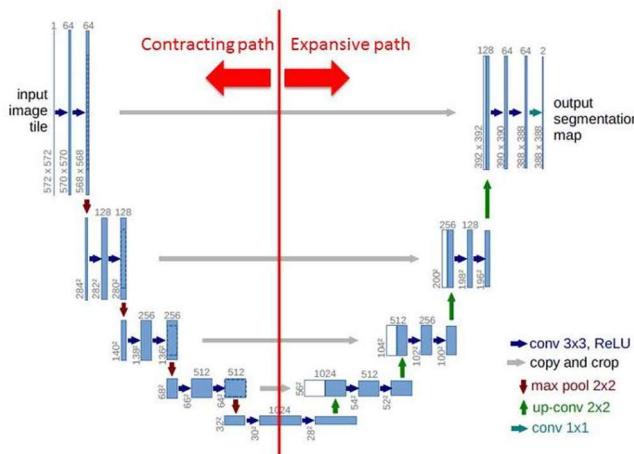
- در حالی که AG ها به خوبی برای وظایف تصویربرداری پزشکی تعمیم می یابند، طراحی و اثربخشی آنها ممکن است نیاز به تنظیم ویژه تسك داشته باشد، و کاربرد مستقیم آن را برای سایر حوزه ها بدون adjustments محدود می کند. به بیان دیگر، هنوز به پیش پردازش کافی و data augmentation برای نتایج بهینه وابسته است.

.۲

(۱)

طراحی معماری مدل U-Net (که در سال ۲۰۱۵ معرفی شد، دارای ساختار U شکل است)

## Network Architecture



U-Net یک معماری شبکه عصبی کانولوشنی است که برای segmentation تصویر طراحی شده است. اجزای اصلی آن عبارتند از:

#### ۱. ساختار رمزگذار-رمزگشای (Encoder-Decoder):

- رمزگذار (Encoder): مسیر انقباض (contracting path) شامل لایه های کانولوشنی است که با فعال سازی ReLU و عملیات max-pooling دنبال می شود. این مسیر اطلاعات contextual را ضبط می کند و تصویر ورودی را در یک بازنمایی ویژگی های فشرده و سطح بالا رمزگذاری می کند.

- مجموعه‌ای از لایه‌های کانولوشنال و max pooling که ابعاد فضایی را به تدریج کاهش می‌دهند.
  - هر block تعداد کانال‌ها را دو برابر می‌کند.
  - ویژگی‌های سلسله مراتبی و اطلاعات زمینه را می‌گیرد. (context information)
  - به طور معمول شامل چندین بلوک از (Conv2D -> ReLU -> Conv2D -> ReLU -> MaxPool) است.
- رمزنگار (Decoder): مسیر گسترش (expansive path)، نمایش فشرده را برای بازسازی نقشه segmentation، نمونه برداری می‌کند. از transposed convolutions یا لایه‌های upsampling یا دنبال آن لایه‌های کانولوشن برای بازیابی وضوح فضایی (spatial resolution) استفاده می‌کند.
- مجموعه‌ای از لایه‌های convolutional و upsampling که ابعاد فضایی را بازیابی می‌کند.
  - هر block تعداد کانال‌ها را به نصف کاهش می‌دهد.
  - اطلاعات مکانی را بازسازی می‌کند و نقشه segmentation را تولید می‌کند.
  - معمولاً شامل (Upsample -> Conv2D -> ReLU -> Conv2D -> ReLU) است.

## :Skip Connections .۲

Skip Connection بالا را از رمزنگار به رمزنگار در همان مقیاس فضایی منتقل می‌کند. آنها ویژگی‌های با وضوحconcatenate feature maps from encoder to (decoder)

نقش: اتصالات پرش (Skip Connections)، جزئیات ریزدانه (fine-grained details) از دست رفته را در حين downsampling در رمزنگار حفظ می‌کند و محلی سازی دقیق (precise localization) را در نقشه segmentation امکان پذیر می‌کند.

- حفظ جزئیات فضایی دقیق از دست رفته در طول downsampling
- به جریان گرادیان در طول آموزش کمک می‌کند
- محلی سازی دقیق را با حفظ context فعال می‌کند
- ویژگی‌های سطح پایین و سطح بالا را ترکیب می‌کند.

اثربخشی و کمک به وظیفه بخش بندی تصویر:

رمزنگار global context را capture می‌کند، در حالی که رمزنگار جزئیات فضایی را بازسازی می‌کند. Skip Connection تضمین می‌کند که جزئیات دقیق حفظ می‌شوند، محلی سازی و زمینه را متوازن می‌کند (balancing)، که برای وظایف segmentation شامل مرزهای پیچیده یا اشیاء کوچک بسیار مهم است.

## (ب) مقایسه با شبکه‌های کاملاً کانولوشنی (FCN)

اصلی‌ترین چیزی که U-Net را متمایز می‌کند استفاده از Skip Connections است. در ادامه توضیح مفصل‌تری آمده است:

1. ویژگی‌های متمایز کننده:

- FCN: ورودی را با استفاده از لایه‌های کاملاً کانولوشنی پردازش می‌کند تا یک نقشه پیش‌بینی متراکم (dense prediction) تولید کند، که معمولاً برای بازیابی وضوح به لایه‌های upsampling مانند bilinear interpolation یا transposed convolution تکیه می‌کند.
- U-Net: دارای ساختار رمزنگار-رمزنگار متقاضی با اتصالات پرش (skip connections) است که به صراحت ضرر جزئیات مکانی در حين downsampling را رفع می‌کند و در نتیجه برای تسک segmentation خیلی بهتر عمل می‌کند.

## ۲. مزایای U-Net نسبت به FCN

Skip :(Preserving Spatial Features & Reducing Information Bottlenecks) -  
 حفظ جزئیات بهتر در U-Net به حفظ جزئیات دقیق کمک می کند و آن را برای تسکهای segmentation دقیق دارند بزرتر می کند. (Precise localization due to skip connections)

### U-Net: Skip Connections

#### Skip Connections:

- Connect encoder layers to decoder layers at the same depth.
- Preserve high-resolution features for better upsampling.

#### Cropping for Alignment:

- Due to valid padding, feature maps in the encoder are larger than those in the decoder.
- Cropping aligns sizes before concatenation in skip connections.

#### Why Important?:

- Combines local details (from encoder) with context (from decoder).
- Helps in precise pixel-wise segmentation.
- Overcomes vanishing gradient issue by allowing smooth information flow.

موثر برای مجموعه دادههای کوچک: طراحی U-Net یادگیری کارآمد را حقیقتاً با تعداد محدودی از نمونههای آموزشی با استفاده از data augmentation و استفاده کارآمد از پارامترها ممکن می سازد. (Better performance with limited training data)

### Data Efficiency in U-Net

- **Residual and Skip Connections:** These connections reduce the amount of information the model needs to learn, as many spatial details are preserved, requiring less adaptation from the model. The need to learn detailed spatial arrangements is mitigated by direct concatenation, which efficiently reintroduces spatial information.
- **Fully Convolutional Nature:** U-Net uses only convolutional layers, which have fewer parameters compared to fully connected networks. This allows the model to generalize well from smaller datasets.
- **Data Augmentation Advantages:** Because U-Net is often used for pixel-wise tasks like semantic segmentation, every augmentation (rotation, scaling, etc.) maintains a one-to-one correspondence with ground truth masks. This allows each augmentation to act as a "new" data point, further enriching the dataset without additional manual labeling.
- **Patch-Based Training:** Instead of processing the entire image (e.g., a 1024x1024 image), U-Net can work with smaller patches, increasing batch sizes and enabling more efficient use of each image during training.

Skip connections :Improving Network Stability and Learning -  
 به گردیدان ها اجازه می دهد تا به طور موثرتری در طول backpropagation رایج در شبکه های عمیق را کاهش دهند.

## ۳. معایب U-Net

هزینه محاسباتی بالاتر: معماری متقارن و skip connections باعث افزایش حافظه و نیازهای محاسباتی می شود. (Computationally intensive for large inputs.)  
حساسیت به اندازه ورودی: معماری U-Net ممکن است نیاز به تغییراتی برای ورودی هایی داشته باشد که بر pooling قابل تقسیم نیستند.  
فرآیند آموزشی پیچیده تری دارد.

## (ب) نقش توابع ضرر (Loss Functions) در آموزش U-Net

تابع ضرر نقش مهمی در هدایت U-Net برای یادگیری دقیق پیش‌بینی‌های سطح پیکسل (pixel-level predictions) دارد. با انتخاب درست تابع ضرر segmentation performance بینه می‌شود.

### ۱. مقایسه :Cross-Entropy Loss and Dice Loss

#### :Cross-Entropy Loss

- خطای طبقه بندی پیکسلی را اندازه گیری می‌کند (pixel-wise classification error).
- زمانی موثر است که توزیع کلاس متعادل باشد.
- برای segmentation چند کلاسه مناسب است.

با مجموعه داده‌های نامتعادل دست و پنجه نرم می‌کند (Sensitive to class imbalance)، زیرا اندازه region را در نظر نمی‌گیرد. همچنین، ممکن است مستقیماً برای کیفیت segmentation نهایه سازی نشود.

#### :Dice Loss

- همپوشانی بین ماسک‌های segmentation پیش‌بینی شده و واقعی را اندازه گیری می‌کند. (overlap between predicted and ground truth segmentation masks)
- بر شباهت در سطح region تمرکز می‌کند و آن را در برابر عدم تعادل class قوی می‌کند (robust to class imbalance).

به طور مستقیم برای کیفیت segmentation بینه می‌شود.  
حساسیت کمتری نسبت به انتخاب آستانه (threshold) دارد.  
به ویژه برای کارهای تصویربرداری پزشکی که در آن مناطق خاص (مانند تومورها) ممکن است در مقایسه با پس زمینه کوچک باشند مفید است.  
ممکن است با اشیاء بسیار کوچک دست و پنجه نرم کند. می‌تواند در طول آموزش اولیه ناپایدار باشد.

### ۲. اولویت و ترجیح دادن:

- از Cross-Entropy Loss تسکهایی که نیاز به دقت پیکسلی (requiring pixel-wise precision) دارند استفاده می‌کنیم.
- از Dice Loss برای مجموعه داده‌های نامتعادل (imbalanced datasets) یا با زمانی که همپوشانی binary segmentation (Focusing on region-based metrics) region-level استفاده می‌کنیم.

### ۳. تابع ضرر ترکیبی پیشنهادی:

$$Loss = \alpha * Cross\ Entropy\ Loss + \beta * Dice\ Loss$$

- همچنین میتوانیم  $\alpha = 1 - \beta$  در نظر بگیریم تا مجموع ضرایب برابر با یک شوند.
- این ترکیب خطی تابع Loss، از دقت پیکسلی pixel-wise precision of Cross-Entropy Loss (Cross-Entropy Loss) برای استحکام region-level robustness of Dice Loss (Dice Loss) بهره می‌برد.
- تنظیم  $\alpha$  و  $\beta$  سهم هر عبارت را متعادل می‌کند و آن را برای تسك های خاص سازگار می‌کند. با توجه مورد نظر و اولویت های مسئله ضرایب را انتخاب می‌کنیم. (Can be adjusted through parameters for specific needs)
- برای سناریوهای segmentation مختلف robust تر است.
- ثبات CE را با بهینه سازی همپوشانی Dice ترکیب می‌کند.

(ت) مدیریت segmentation چند کلاسه در U-Net

۱. رویکرد مورد استفاده برای وظایف segmentation چند کلاسه:

- لایه خروجی: از یک تابع فعال سازی softmax در لایه نهایی برای تولید توزیع احتمال برای هر پیکسل در چندین کلاس استفاده کنید.

- تابع ضرر: یا یک Dice Loss (categorical cross entropy) یا یک multi-class Cross-Entropy تعیین یافته که میتواند برای کلاس های متعدد اعمال شود.

رویکرد پیاده سازی:

- Output channel per class در لایه نهایی

- تابع فعال سازی Softmax برای توزیع احتمالات کلاس ها در آخرین لایه

- (ground truth labels) برچسب های حقیقی One-hot encoding

۲. چالش ها:

عدم تعادل کلاس ها (Class Imbalance): کلاس های خاصی ممکن است غالب باشند که باعث می شود مدل کلاس های اقلیت را نادیده بگیرد. (poor performance for minority classes)

راه حل:

- Weighted loss functions از وزن دهنده کلاس ها در تابع ضرر استفاده کنیم تا اهمیت بالاتری را به کلاس های کمتر ارائه شده اند اختصاص دهیم.

- Class-wise data augmentation کلاس های اقلیت را بیشتر augmentation کنیم.

- Focal loss implementation از تابع ضرر Focal loss استفاده کنیم.

- Balanced sampling strategies نمونه برداری را طوری تنظیم کنیم تا تعادل بیشتر حفظ شود.

مرزهای مبهم (Ambiguous Boundaries): کلاس های همسایه (Neighboring classes) ممکن است مرزهای نامشخصی (label ambiguity) داشته باشند. (unclear boundaries)

راه حل:

برای اصلاح پیش‌بینی‌ها، از توابع ضرر boundary-aware (به عنوان مثال، IoU) یا تکنیک‌های پس پردازش مانند Conditional Random Fields (CRFs) استفاده کنیم. (Post-processing with CRF)

- Boundary-aware loss functions (تابع ضرر آگاه از مرز)

- Multi-scale feature fusion (ترکیب ویژگی چند مقیاسی)

- Attention mechanisms to focus on boundary regions با استفاده از مکانیزم توجه به مرزها بیشتر توجه کنیم.

این راه حل‌های جامع، U-Net را به یک معماری قدرتمند و انعطاف‌پذیر برای وظایف segmentation مختلف و چند کلاسه تبدیل می‌کند، هرچند برای بهینه‌سازی عملکرد آن، بررسی دقیق ترکیب خاص مورد نیاز است.

(ث) بخش اول: ابعاد فضای ویژگی در عمیق ترین لایه

فرض میکنیم معماری U-Net ابعاد را در هر لایه رمزگذار(encoder) به نصف کاهش می دهد و در لایه رمزگشای(decoder) دو برابر می کند. هم چنین فرض میکنیم که در این معماری U-Net ۴ مرحله downsampling در رمزگذار انجام میشود. با توجه به ابعاد تصویر ورودی که  $256 * 256$  است:

:Encoder . ۱

- ابعاد در ابتدا (لایه اول):  $256 * 256$
- بعد از لایه اول encoder (لایه دوم):  $128 * 128$  (به نصف کاهش می یابد)
- بعد از لایه دوم encoder (لایه سوم):  $64 * 64$  (به نصف کاهش می یابد)
- بعد از لایه سوم encoder (لایه چهارم):  $32 * 32$  (به نصف کاهش می یابد)
- بعد از لایه چهارم encoder (لایه پنجم، عمیق ترین لایه):  $16 * 16$

بنابراین، در عمیق ترین لایه، فضای ویژگی دارای ابعاد  $16 * 16$  است.

۲. بنابراین تعداد کل پیکسل ها  $256$  تا است:

$$16 * 16 = 256 \text{ pixels}$$

بخش دوم: تعداد پارامترها در لایه کانولوشن دوم رمزگذار

با توجه به این که در  $4$  لایه داریم که به ترتیب  $64, 128, 256, 512$  فیلتر دارند به محاسبه می پردازیم.

۱. دومین لایه کانولوشن رمزگذار مربوط به لایه با  $128$  فیلتر است. همچنین داریم:

- هر لایه کانولوشن از کرنل  $3 * 3$  استفاده می کند.
- تعداد کanal های ورودی برابر با تعداد فیلترهای لایه قبلی است که در اینجا برابر با  $64$  است چراکه لایه اول رمزگذار  $64$  تا فیلتر دارد.

۲. تعداد پارامترها: برای یک لایه کانولوشنی داریم:

Parameters = (Kernel Width  $\times$  Kernel Height  $\times$  Input Channels + Bias)  $\times$  Output Filters.

فیلترهای خروجی \* (بایاس + کanal های ورودی \* ارتفاع هسته \* عرض هسته) = پارامترها

در این مثال:

- اندازه کرنل  $= 3 * 3$
- کanal های ورودی  $= 64$  (از لایه قبلی)
- فیلترهای خروجی  $= 128$
- بایاس  $= 1$  در هر فیلتر (در مجموع  $128$ )

جایگزینی مقادیر چنین می شود:

$$\text{Tعداد پارامترها} = (3 * 3 * 64 + 1) * 128$$

۳. ساده سازی:

$$\text{Tعداد پارامترها} = (576 + 1) * 128 = 577 * 128 = 73,856$$

بنابراین، دومین لایه کانولوشن رمزگذار دارای  $73,856$  پارامتر است.

## سوال چهارم

### ۱. چگونگی تقسیم داده ها

برای این سیستم تبلیغاتی، داده ها باید با در نظر گرفتن چندین عامل کلیدی به دقت تقسیم شوند:

#### ۱. تقسیم زمانی (Temporal Splitting):

- با توجه به اینکه الگوهای تبلیغاتی و رفتار کاربر در طول زمان تغیر می کند، به جای تقسیم تصادفی از تقسیم بندی مبتنی بر زمان استفاده می کنیم. در واقع، از آنجایی که عملکرد تبلیغات ممکن است در طول زمان متفاوت باشد (به عنوان مثال، به دلیل اثرات یا روندهای فصلی)، داده ها را بر اساس زمان تقسیم می کنیم.
- آخرین و جدیدترین داده ها باید به مجموعه Test، داده های کمی قدیمی تر به مجموعه Dev و قدیمی ترین داده ها به مجموعه Train می روند.
- برای جلوگیری از نشت زمانی (time leakage)، بر اساس week\_id تقسیم می کنیم. با داده های week یا هفته های قدیمی تر آموزش می دهیم و با داده های هفته های جدیدتر مدل را تست می کنیم.
- مثال: از داده های هفته ۱ تا ۶ برای Train-Dev، هفته ۷ برای Train-Dev، هفته ۸ برای Dev و هفته ۹ برای Test استفاده کنیم. این تقسیم بندی شبیه سازی نحوه استفاده از مدل در production است، جایی که مدل نیاز به پیش بینی عملکرد آینده دارد.

#### ۲. ملاحظات Distribution:

- Train و Train-Dev باید از یک دوره زمانی مشابه، برای اطمینان از مشابه توزیع ها باشند.
- مجموعه های Dev و Test باید از نظر زمانی به یکدیگر نزدیک تر باشند تا توزیع داده های فعلی را نشان دهند.
- اطمینان حاصل کنیم که هر split دارای نمونه های کافی از هر دو نوع تبلیغات کلیکی و Action است.

#### ۳. ملاحظات کلیدی:

- جداسازی موجودیت (Entity Separation): اطمینان حاصل کنیم که تبلیغات / کمپین های جدید در Dev/Test برای test generalization در Train نیستند.
- طبقه بندی (Stratification): تعادل طبقاتی (به عنوان مثال، انواع تبلیغات، کمپین ها) را در بین تقسیم بندی ها حفظ کنیم. به عبارت دیگر، اطمینان حاصل کنیم که هر زیرمجموعه توزیع یکسانی از ویژگی های کلیدی، مانند type (کلیک یا اکشن) را برای جلوگیری از bias حفظ می کند.
- اجتناب از نشت (Leakage): ویژگی های تحت تأثیر رویدادهای آینده را حذف کنیم (به عنوان مثال، داده های پس از کلیک در ویژگی های قبل از کلیک).
- بر زدن داده ها (Data Shuffling): قبل از splitting، داده های خود را به طور تصادفی به هم بزنیم تا مطمئن شویم که هر زیرمجموعه نماینده مجموعه داده کلی است.

#### ۴. نسبت اندازه ها (Size Ratios):

- Train: داده های کافی برای یادگیری الگوها داشته باشیم: بزرگترین بخش به عنوان مثال ۷۰٪ برای آموزش مدل استفاده می شود. همچنین اطمینان حاصل کنیم که شامل نمونه های متنوعی است (به عنوان مثال، تبلیغات از کمپین های مختلف، موقعیت ها و هفته ها).
- Train-Dev: تعمیم مدل را در توزیع آموزش ارزیابی کنیم: زیر مجموعه کوچک به عنوان مثال ۱۰-۱۵٪ از توزیع مشابه داده های آموزشی. برای تشخیص overfitting استفاده می شود (به عنوان مثال، اگر مدل در آموزش خوب عمل کند اما در اینجا ضعیف باشد، نشان دهنده واریانس بالا است).
- Dev: هایپرپارامترها را تنظیم کنیم و performance را ارزیابی کنیم: به طور ایده آل منعکس کننده توزیع هدف (به عنوان مثال، هفته های آینده یا تبلیغ کنندگان / کمپین های جدید). برای تنظیم هایپرپارامتر و تشخیص عدم تطابق داده ها (data mismatch) استفاده می شود.
- Test: ارزیابی نهایی را انجام دهیم: از توزیع مشابه مجموعه Dev. برای ارزیابی نهایی برای جلوگیری از overfitting با مجموعه Dev استفاده می شود.

## ۲ تحلیل خطای آموزش

### ۱. تحلیل سناریو:

#### الف) خطای آموزش بالا (High Training Error)

مشکل: مدل underfitting (high bias) دارد و فاقد ظرفیت کافی برای یادگیری الگوهای اساسی در داده ها بسیار ساده است.

راه حل:

- معناری های مدل پیچیده تری را امتحان کنیم. مثلا با افزودن لایه ها یا نورون های بیشتر.
- ویژگی های مرتبط بیشتری را اضافه کنیم.
- از معناری قدرتمندتری استفاده کنیم.
- در صورت وجود regularization را کاهش دهیم.

#### ب) خطای آموزش کم اما خطای Train-Dev بالا:

مشکل: مدل overfitting به داده های آموزشی دارد (واریانس بالا) و به خوبی تعمیم نمی دهد.

راه حل:

- افزایش regularization (به عنوان مثال، L1، L2)
- اجرای dropout
- پیچیدگی و اندازه مدل را کاهش دهیم.
- داده های آموزشی بیشتری اضافه کنیم.
- از Data Augmentation استفاده کنیم.

#### ج) خطای پایین Train-Dev و Train اما خطای بالای Dev

مشکل: عدم تطابق توزیع (Distribution mismatch) بین داده های آموزشی (Train-Dev و Train) و داده های فعلی (Dev)

راه حل:

- تراز کردن داده های آموزشی با توزیع Dev (به عنوان مثال، جمع آوری داده های مشابه، تطبیق دامنه)
- به روز رسانی داده های آموزشی برای گنجاندن نمونه های جدیدتر
- پیاده سازی تکنیک های domain adaptation
- نرمال سازی ویژگی (feature normalization) در بازه های زمانی را در نظر بگیریم.
- داده ها را مجدداً split کنیم تا Train و Dev بیشتر representative مجموعه Test شوند.
- تنوع داده های آموزشی را افزایش دهیم.

#### د) خطای پایین Dev اما خطای Test بالا:

مشکل: مدل از طریق تنظیم های پارامتر به مجموعه Dev بیش از حد برازش (overfitting to Dev) می شود. (overfitting to Dev set through hyperparameter tuning)

راه حل:

- داده های مجموعه Dev بیشتری دریافت کنیم.
- تعداد تکرارهای تنظیم های پارامتر (hyperparameter tuning) را کاهش دهیم.
- مطمئن شویم که مجموعه های Dev و Test از دوره های زمانی مشابهی (similar time periods) هستند.
- مجموعه های Dev و Test بزرگ تر نیز ممکن است کم کنند.
- ممکن است هر دو مجموعه از یک توزیع گرفته نشده باشند. در این صورت، مجموعه های Dev و Test خود را مجدداً تعریف کنیم تا داده های را که مدل در production با آن مواجه می شود بهتر نشان دهیم.

۲. خیر، به احتمال زیاد بی اثر است. افزایش سایز داده های آموزشی به تنها کمک به خطاهای آموزشی بالا (High Training Error) نمی کند. از آنجایی که مدل دارای مشکل underfitting یا high bias نشان می دهد)، مسئله اصلی ناتوانی مدل در یادگرفتن الگوها در داده های موجود است. افزودن داده های بیشتر کمک نمی کند تا زمانی که ظرفیت یادگیری مدل در ابتدا بهبود یابد.

در واقع، Underfitting از محدودیت های مدل ناشی می شود، نه کمیت داده. بنابراین باید روی بهبود ظرفیت/ویژگی های مدل تمرکز کنیم.

۳. اختلاف قابل توجه مدل در پیش بینی و درآمد واقعی: عوامل متعددی می توانند باعث این امر شود:

- عدم تطابق متریک بهینه سازی (Metric Misalignment Optimization Metric Mismatch): مدل ها برای پیش بینی cross-entropy accuracy (مثلاً  $\text{bid} \times \text{CTR} \times \text{CVR}$ ) و نه برای پیش بینی درآمد (high-value) بهینه شده اند.
- تأثیر Bid Value: پیش بینی های با accuracy بالا ممکن است با تبدیل های با ارزش بالا (high-value) مرتبط باشد.
- سوگیری موقعیت (Position Bias): پیش بینی های CTR/CVR ممکن است اثرات خاص position را در نظر نگیرند. در واقع، مدل ها ممکن است اثرات موقعیت را نادیده بگیرند (به عنوان مثال، تبلیغات در top positions بدون در نظر گرفتن ارتباط، کلیک های بیشتری دریافت می کنند).
- اثرات زمانی (Temporal Effects): الگوهای درآمد ممکن است سریعتر از الگوهای کلیک/تبدیل تغییر کنند.
- در پیش بینی ها: برآورد بیش از حد زیاد یا برآورد بیش از حد کم سیستماتیک CTR/CVR برای تبلیغات با مقدار زیاد bid.
- انتشار خطای خطاهای کوچک CTR/CVR در محاسبات درآمد ترکیب می شوند (به عنوان مثال،  $10\% \text{ CTR error} \times \text{high bid} \rightarrow \text{large revenue loss}$ ).
- فرضیات نادرست در فرمول های Click یا Interaction.
- Skewed distributions: به عنوان مثال، انواع خاصی از تبلیغات بر درآمد غالب است و اثرات context (مانند کاربر، نوع دستگاه) را نمی توانند مدل مشخص نشده است (not captured by the model)، مانند تغییرات رفتار کاربر، شرایط بازار خارجی، یا عدم دقت در مقادیر bid.

رویکردهای راه حل این مشکل:

- با استفاده از یادگیری تقویتی یا تابع custom loss، به طور مستقیم مدل را برای درآمد بهینه سازی کنیم.
- توابع revenue-weighted loss را استفاده کنیم.
- bid value را به عنوان یک ویژگی در مدل ها اضافه کنیم.
- تصحیح positional bias را بگنجانیم. مدل های جداگانه برای محدوده position های مختلف ایجاد کنیم یا یک مدل آگاه از position را آموزش دهیم.
- مدل ها را با داده های اخیر (recent data) بیشتر، به روز کنیم.
- خروجی های مدل را کالیبره کنیم تا مطمئن شویم که احتمالات CTR/CVR منعکس کننده نرخ های واقعی هستند.
- مفروضات ساخته شده در طول مدل سازی را مرور کنیم.
- اطمینان حاصل کنیم که همه ویژگی های مربوطه گنجانده شده است، و عوامل خارجی را که ممکن است بر درآمد تأثیر بگذارد در نظر بگیریم.
- تست کردن اختلاف بین نتایج پیش بینی شده و مشاهده شده در بخش های مختلف (مانند کمپین ها، موقعیت ها (positions)).

### ۳ استفاده از انتقال یادگیری

۱. به کار بردن انتقال یادگیری: انتقال یادگیری زمانی مناسب است که:

- وظایف منبع و هدف (source and target tasks)، فضاهای ویژگی مشابهی را به اشتراک می‌گذارند.
- داده‌های کافی برای تسک target وجود ندارد.
- تسک source دارای داده‌های فراوان است.
- الگوهای اساسی وجود دارد که می‌تواند بین وظایف منتقل شود.

در مورد این مسئله، انتقال یادگیری بسیار کاربردی است زیرا:

- پیش‌بینی CTR داده‌های فراوانی دارد.
- پیش‌بینی CVR فاقد نمونه‌های مثبت کافی است.
- هر دو تسک شامل الگوهای رفتاری مشابه کاربر (similar user behavior patterns) هستند.

به طور خلاصه: انتقال یادگیری زمانی مناسب است که یک تسک source (به عنوان مثال، پیش‌بینی CTR) دارای داده‌های فراوان باشد، و تسک target (پیش‌بینی CVR) داده‌های محدودی داشته باشد. الگوهای آموخته شده (به عنوان مثال، رفتار کاربر، محتوای تبلیغات) را از source برای بهبود تسک target انتقال می‌دهیم. (از آنجایی که هر دو تسک شامل تعامل کاربر با تبلیغات است، دانش از پیش‌بینی CTR می‌تواند به پیش‌بینی CVR منتقل شود.)

کاربرد:

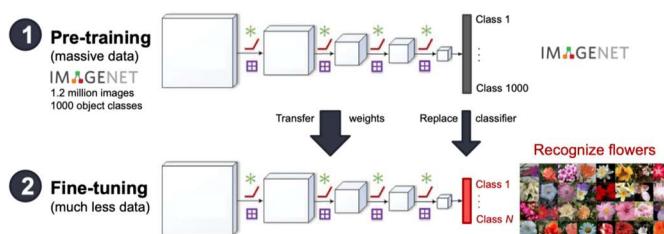
- در تسکی که داده‌های برچسب دار کافی وجود دارد، مدل را روی پیش‌بینی CTR از Pretrain می‌کنیم.
- در تسک CVR را در تسک pretrained model با استفاده از داده‌های محدود Fine-tune می‌کنیم.

در ادامه جدولی کلی در خصوص استفاده از انتقال یادگیری به صورت عمومی با استفاده از pretrained model آمده است:

	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers or start from scratch!

#### Transfer Learning

- ① Pretrain on a large dataset (e.g. ImageNet)
- ② Fine-tune it on a small target dataset (e.g. Flowers)



۲. پیشنهاد تسک جایگزین: از پیش بینی CTR به عنوان source task استفاده میکنیم (پیش بینی CTR به عنوان تسک جایگزین) زیرا داده های CTR فراوان است. هر دو تسک شامل تعامل کاربر پس از نمایش (user engagement post-impression) است، بنابراین ویژگی های مشترک (به عنوان مثال، ارتباط آگهی تبلیغاتی، demographics کاربر) قابل انتقال است. یک مدل از پیش آموزش دیده (pretrained model) با CTR را روی داده های CVR تنظیم دقیق (Fine-tune) میکنیم. دلایل انتخاب در ادامه به شکل مفصل تر آمده است:

- رفتار کلیک اغلب با احتمال تبدیل مرتبط است (CTR و CVR همبستگی دارند).
- پیش بینی CTR یک تسک بسیار مرتبط است که در آن داده ها فراوان تر است. (CTR داده های بسیار بیشتری در دسترس دارد).
- هر دو تسک دارای فضاهای ویژگی مشابه (ویژگی های تبلیغات، موقعیت وغیره) هستند.
- Embedding های آگهی و position از پیش بینی CTR برای پیش بینی CVR ارزشمند است.
- یک مدل pretrained در مورد پیش بینی CTR ویژگی های مشترک مانند رفتار کاربر، اثرات موقعیت و ارتباط تبلیغات را یاد می گیرد که برای پیش بینی CVR بسیار مفید است.

۳. Multi-Task Learning: بله، Multi-Task Learning در اینجا بسیار مفید خواهد بود. به طور مشترک چندین تسک را آموزش می دهد و representation ها را به اشتراک می گذارد.

مزایا:

- کارایی داده (Data Efficiency): مجموعه داده های بزرگتر CTR و سایر وظایف کمک، پراکنده ای داده CVR را جبران می کند. در واقع، با استفاده از داده های فراوان CTR و سایر وظایف کمک به مشکل عدم تعادل داده ها کمک می کند.
- یادگیری representation مشترک بین پیش بینی CTR و CVR (اشتراک گذاری ویژگی Feature Sharing) یا (adv\_id, ad\_id). وظایف از ویژگی های مشترک تبلیغ و کاربر (مثلًا جاسازی های (multiple related tasks)).
- اثر Regularization از یادگیری چندین تسک مرتبط (regularization).
- استفاده کارآمدتر از داده های تبدیل محدود.
- تعمیم بهبود یافته از طریق وظایف کمک (auxiliary tasks). دانش یا لایه های مشترک از overfitting داده های پراکنده CVR (sparse CVR data) جلوگیری می کند.

وظایف پیشنهادی:

1. وظایف اصلی (Main tasks):
  - پیش بینی CVR (مسئله اصلی این قسمت سوال که با کمبود داده های مثبت مواجه است).
  - پیش بینی CTR
2. وظایف کمکی (Auxiliary tasks):
  - پیش بینی زمان مشاهده (View-time)
  - پیش بینی تعامل پس از کلیک (Post-click engagement)
  - پیش بینی تعامل مبتنی بر موقعیت (Position-based interaction)
  - پیش بینی معیارهای تعامل کاربر (به عنوان مثال، زمان ماندن).
  - پیش بینی زمان click-to-action (برای مدل سازی (sequential tasks).

پیاده سازی:

- معماری: لایه های مخفی مشترک (Shared hidden layers) با سرهای خروجی خاص (task-specific output heads).
- لایه های مشترک (Shared layers): ویژگی های مشترک (مانند آگهی، موقعیت، کاربر) را در بین وظایف میاموزیم.
- سرهای خاص تسک (Task-specific heads): لایه های خروجی را برای پیش بینی CVR، CTR و سایر وظایف کمک (auxiliary tasks) جدا کنیم.
- تابع ضرر: ضرر ها را ترکیب کنیم (به عنوان مثال، مجموع وزن دار توابع ضرر CTR و CVR).

این وظایف الگوهای اساسی تعامل کاربر را به اشتراک می گذارند و می توانند به عملکرد کلی بهتر و generalization بهتر کمک کنند.

نتیجه گیری:

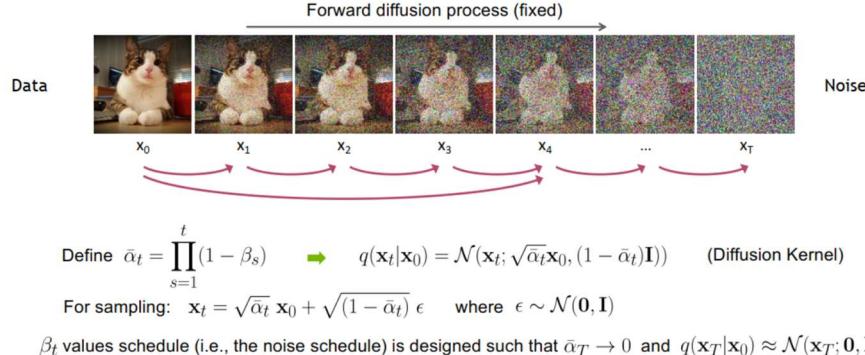
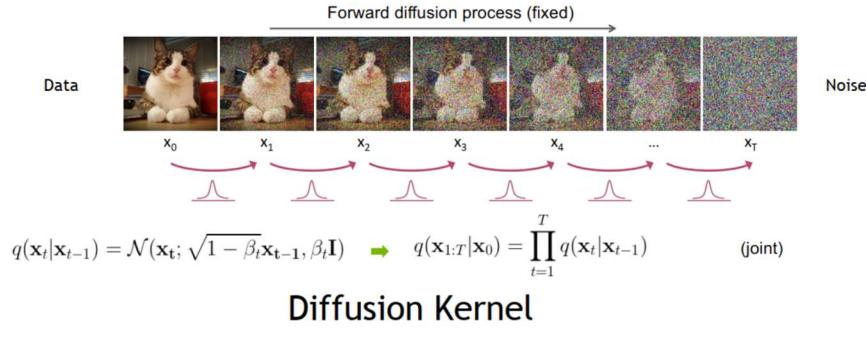
Multi-Task Learning و Transfer Learning هر دو حل های قابل اجرا هستند. Multi-Task Learning مزیت افزوده بهینه سازی مشترک برای وظایف مرتبط را ارائه می دهد.

## سوال پنجم (۱)

ابتدا مطالب زیر که در اسلاید های درس بود را به عنوان پیش نیاز و فرض برای حل مسئله در نظر بگیرید:

## Forward Diffusion Process

The formal definition of the forward process in T steps:



حال که مسئله را به خوبی متوجه شدیم به اثبات می پردازیم:

هدف این است که نشان دهیم مرحله میانی  $x_t$  را می توان مستقیماً از  $x_0$  با استفاده از ویژگی های نویز گاوی، بدون اضافه کردن نویز تکراری، نمونه برداری کرد.

از زنجیره مارکوف برای فرآیند انتشار به جلو میدانیم:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I)$$

٩

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1})$$

### بینش کلیدی برای اثبات: ترکیب توزیع های گاوی

اضافه شدن نویز گاوی در هر مرحله، در یک term نویز گاوی منفرد (single Gaussian noise term) انباسته می شود. بیاید  $x_t$  را بر حسب  $x_0$  برای  $t > 1$  بنویسیم.

مرحله ۱ : برای  $t = 1$

$$x_1 = \sqrt{1 - \beta_1} x_0 + \sqrt{\beta_1} \epsilon_1, \quad \epsilon_1 \sim \mathcal{N}(0, I)$$

مرحله ۲ : برای  $t = 2$

$$x_2 = \sqrt{1 - \beta_2} x_1 + \sqrt{\beta_2} \epsilon_2, \quad \epsilon_2 \sim \mathcal{N}(0, I)$$

جایگزینی  $x_1$  به  $x_2$ :

$$x_2 = \sqrt{1 - \beta_2} (\sqrt{1 - \beta_1} x_0 + \sqrt{\beta_1} \epsilon_1) + \sqrt{\beta_2} \epsilon_2$$

گسترش این عبارت،

$$x_2 = \sqrt{(1 - \beta_2)(1 - \beta_1)} x_0 + \sqrt{(1 - \beta_2)\beta_1} \epsilon_1 + \sqrt{\beta_2} \epsilon_2$$

با استفاده از Reparameterization Trick داریم:

$$x_2 = \sqrt{(1 - \beta_2)(1 - \beta_1)} x_0 + \sqrt{1 - (1 - \beta_2)(1 - \beta_1)} \epsilon$$

تعمیم دهنده:

برای  $t > 2$ ، با جایگزینی بازگشتی (recursively substituting) برای مقادیر  $x$  قبلی، عبارت به صورت زیری شود:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \epsilon$$

که در آن:

$$\bar{\alpha}_t = \prod_{i=1}^t (1 - \beta_i), \quad \epsilon \sim \mathcal{N}(0, I)$$

بنابراین،  $x_t$  را می توان مستقیماً از  $x_0$  به صورت زیر نمونه برداری کرد:

$$q(x_t | x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)I)$$

این ثابت می کند که نویز را می توان در یک مرحله اضافه کرد.

همچنین اثبات دقیق تر و با جزییات بیشتر در این لینک (<https://codoraven.com/blog/ai/diffusion-model-clearly-explained/>) هست که تصویر آن به شرح زیر است:

$$\begin{aligned} x_t &= \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \varepsilon_{t-1} & \varepsilon_0, \dots, \varepsilon_{t-2}, \varepsilon_{t-1} &\sim \mathcal{N}(0, I) \\ &= \sqrt{\alpha_t} \boxed{x_{t-1}} + \sqrt{1 - \alpha_t} \varepsilon_{t-1} & \bar{\varepsilon}_0, \dots, \bar{\varepsilon}_{t-2}, \bar{\varepsilon}_{t-1} &\sim \mathcal{N}(0, I) \\ &= \sqrt{\alpha_t} \left( \sqrt{\alpha_{t-1}} x_{t-2} + \sqrt{1 - \alpha_{t-1}} \varepsilon_{t-2} \right) + \sqrt{1 - \alpha_t} \varepsilon_{t-1} & \alpha_t = 1 - \beta_t \\ &= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \boxed{\sqrt{\alpha_t(1 - \alpha_{t-1})} \varepsilon_{t-2}} + \boxed{\sqrt{1 - \alpha_t} \varepsilon_{t-1}} & \bar{\alpha}_t = \prod_{i=1}^t \alpha_i \\ && X + Y \end{aligned}$$

Reparameterization Trick

Underlying Normal Distribution

$$\begin{aligned} 0 + \sqrt{\alpha_t(1 - \alpha_{t-1})} \varepsilon_{t-2} &\xrightarrow{\text{-----}} X \sim \mathcal{N}(0, \alpha_t(1 - \alpha_{t-1}) I) \\ 0 + \sqrt{1 - \alpha_t} \varepsilon_{t-1} &\xrightarrow{\text{-----}} Y \sim \mathcal{N}(0, (1 - \alpha_t) I) \end{aligned}$$

Recall:

$$\text{If } X \sim \mathcal{N}(\mu_X, \sigma_X^2) \quad Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$$

$$Z = X + Y$$

$$\text{Then } Z \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$$

$$Z \sim \mathcal{N}(0, \boxed{\sigma_X^2 + \sigma_Y^2})$$

$$Z \sim \mathcal{N}(0, \boxed{(1 - \alpha_t \alpha_{t-1})} I)$$

Reparameterization Trick

$$\sigma_X^2 + \sigma_Y^2 = \alpha_t(1 - \alpha_{t-1}) + (1 - \alpha_t)$$

$$= \cancel{\alpha_t} - \alpha_t \alpha_{t-1} + 1 - \cancel{\alpha_t}$$

$$= 1 - \alpha_t \alpha_{t-1}$$

$$0 + \sqrt{1 - \alpha_t \alpha_{t-1}} \bar{\varepsilon}_{t-2}$$

$$= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \boxed{\sqrt{1 - \alpha_t \alpha_{t-1}} \bar{\varepsilon}_{t-2}}$$

⋮

$$= \sqrt{\alpha_t \alpha_{t-1} \dots \alpha_1} x_0 + \sqrt{1 - \alpha_t \alpha_{t-1} \dots \alpha_1} \varepsilon$$

$$= \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon$$

(ب)

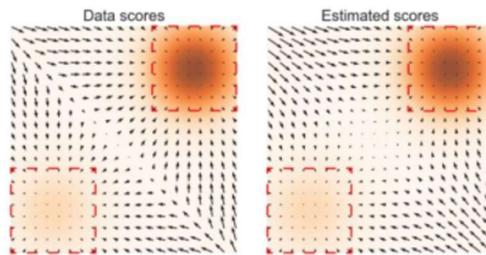
جواب این سوال در اسلاید زیر از درس آمده است. ابتدا آن را ببینیم و سپس توضیح دهیم:

## What are the problems ?

1. No Support (everywhere)  
i.e.  $p(x)$  not defined and thus gradient not defined

→ Solution: Add Gaussian Noise

2. Inaccurate score estimation in low density regions



→ Solution add noise at different magnitudes

(large noise: filling low density regions, small noise: fine-adjustments in high density regions )

مشکل از چند عامل کلیدی ناشی می شود:

Sampling Efficiency (۱): هنگام استفاده ازتابع هدف basic score matching که توضیح داده شد:

$$E_{p(x)}[||\nabla \log(p(x)) - s_\theta(x)||^2]$$

مدل یاد می گیرد که score function (gradient of log density) را فقط در نقاطی که نمونه داده ها را داریم تخمین بزند. این یعنی:

در مناطق کم تراکم دور از نقاط داده واقعی (low-density regions far from real data points)، تخمین score function گیری می شود.

فرآیند نمونه گیری می تواند در این مناطق که تابع امتیاز تخمین ضعیفی دارد گیر کند.

این منجر به خروجی های نویزی می شود زیرا مدل گرادیان های خوبی برای فضای کامل نیاموخته است.

:Scale Sensitivity (۲)

تابع هدف basic score matching با تمام مناطق فضای داده به طور مساوی رفتار می کند، اما در عمل:

- مناطق با تراکم بالا (جایی که بیشتر داده های واقعی در آن قرار دارد) به تخمین گرادیان دقیق تری نیاز دارند.

- مناطق کم تراکم می توانند شبیه های بسیار بزرگ داشته باشند که بر objective غالب کنند.

- این باعث می شود آموزش unstable باشد و می تواند منجر به همگرای ضعیف شود.

راه حل: اضافه کردن نویز (Denoising Score Matching and Noise Conditioning)

راه حل کلیدی، تزریق صحیح نویز در مقیاس های مختلف و آموزش مدل برای حذف نویز در هر مقیاس است. این کار توسط:

۱) اضافه کردن نویز گاوی به نقاط داده:

$$q_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x}|x, \sigma^2 I)$$

: (estimate the score of the noised distribution) نویزی شده توزیع امتیاز تخمین آموزش مدل برای (۲)

$$E_{p(x)q_\sigma(\tilde{x}|x)} \left[ \left\| \nabla \log q_\sigma(\tilde{x}|x) - s_\theta(\tilde{x}, \sigma) \right\|^2 \right]$$

## اين روش مشکلات را حل مي کند زيرا:

: (Better Coverage) پوشش بہتر (۱)

- تزریق نویز تضمین می کند که مدل نقاط داده را در تمام فضا می بیند. (چون توزیع پف میکند)

- مدل گردیدن های معنی دار را حقیقت در مناطق کم تراکم باد می، گردید.

این منجر به مسیرهای نمونه‌گیری باید ادارت می‌شود.

## ۲) برنامه طبعی (Natural Curriculum)

شروع با سطوح بالای نوبن و کاهش، تدریج، آنها یک برنامه طبیع، اتحاد ممکنند.

مدل، ابتدا ساختار درشت را باد م، گهاد سیس، حنثیات را اصلاح م، کند.

- این افراد باعث بایدای، بیشتر آمزش شده و منج به همگاران بهتة ع. شود.

۳) بادگهی، متناسب با مقیاس:

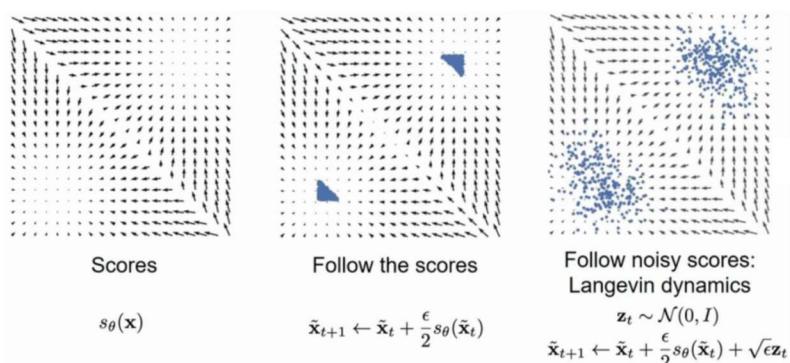
سطوح مختلف، نوبت به طفو طبع، مقابس، های مختلف، را در داده ها کنترل م. کنند.

محله کارخانه های مناسن و اداری، مناطق راهگاهی، راهنمایی و درمانی و مدارس

این راهیں میں ممکن را بروز نہیں پیدا کر سکتے اور پیشیں یہاں تک کہتے ہیں کہ قدرتی ایجاد شدہ

رویکرد افروزن نویز با انحراف معیار های مختلف به طور موثر مشکل دشوار تخمین گرادیان توزیع داده های ناشناخته را به یک سری مسائی، نوبن زدای، به خود تعریف شده در مقابس، های مختلف تبدیل م کنید.

به همین دلیل است که مدل‌های diffusion مدرن از این رویکرد نویز شرطی به جای basic score matching استفاده می‌کنند، زیرا نمونه‌های با کفیت بسیار بالات با نهاد کمتر در خروج‌ها تولید می‌کنند.



## پیان