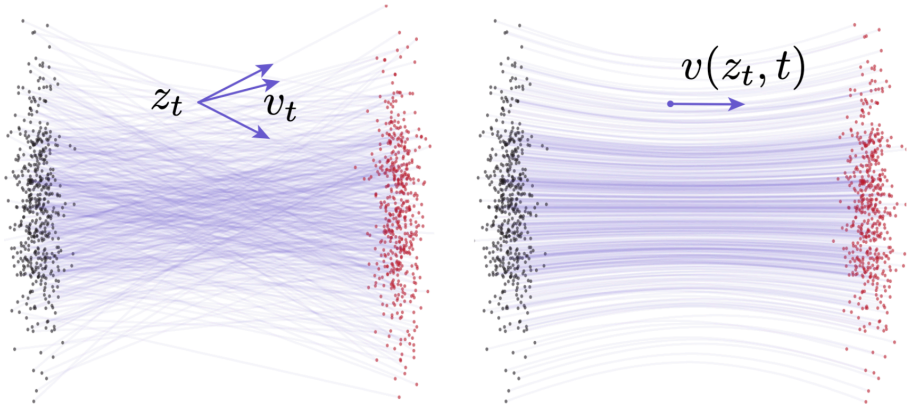
Mean Flows Note

Background: Flow Matching

流匹配是一类生成式模型，它通过学习两个概率分布之间的“流”，也就是速度场，来进行概率分布的转换。形式上，给定数据$x \sim p_{\text{data}}(x)$以及先验分布$\epsilon \sim p_{\text{prior}}(\epsilon)$，可以构造一个随时间$t$变化的流路径$z_t = a_t x + b_t \epsilon$，其中$a_t$和$b_t$是事先定义好的调度器。速度$v_t$定义为$v_t = z_t' = a_t' x + b_t' \epsilon$，其中$'$表示对时间$t$求导。这个速度被称为条件速度(Conditional Velocity)，记为$v_t = v_t(z_t|x)$。一个常用的调度器是$a_t = 1 - t,\ b_t = t$，于是可以推导得到$v_t = \epsilon - x$。



左图为条件流，其中给定的z_t可以由不同的(x, ϵ)对产生，从而导致不同的条件速度v_t。

右图为边际流，通过对所有可能的条件速度进行边缘化得到。边际速度场作为训练神经网络的底层真值场。

由于给定的z_t以及其速度v_t可以由不同的x和ϵ产生，所以流匹配的本质就是对所有可能性的期望进行建模，称为边际速度：

$$v(z_t, t) = \mathbb{E}_{p_t(v_t|z_t)}[v_t]$$

我们通过学习一个以θ为参数的神经网络v_θ来拟合边际速度场，即：

$$\mathcal{L}_{FM}(\theta) = \mathbb{E}_{t, p_t(z_t)} \| v_\theta(z_t, t) - v(z_t, t) \|^2$$

通过前置学习，我们可以用条件流匹配损失来代替边际流匹配损失，即：

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t, x, \epsilon} \| v_\theta(z_t, t) - v(z_t|x) \|^2$$

其中训练目标v_t是条件速度。最小化$\mathcal{L}_{CFM}$与最小化$\mathcal{L}_{FM}$两者完全等价。

给定一个边际速度场$v(z_t, t)$，我们通过求解一个关于z_t的常微分方程(ODE)来生成样本，即：

$$\frac{d}{dt} z_t = v(z_t, t)$$

其初始条件为$z_1 = \epsilon \sim p_{\text{prior}}$，该方程的解可以写为$z_r = z_t - \int_r^t v(z_\tau, \tau) d\tau$，其中$r$表示另外一个时间步。在实际中，这种积分是在离散时间步上进行数值逼近的。例如，一阶常微分方程组求解器Euler方法每一步的计算为：

$$z_{t_{i+1}} = z_{t_i} + (t_{i+1} - t_i) v(z_{t_i}, t_i)$$

值得注意的是，即使条件流被设计成“直的”，边缘速度场也通常会诱导出一条弯曲的轨迹。这种非直观性不仅仅是神经网络逼近的结果，而且是来自于底层真值场。当在湾区轨迹上应用离散化时，数值ODE求解器会导致不准确的结果。

MeanFlow Models

Mean Flows

该方法的核心思想是引入一个表示平均速度的新场，而在流匹配中模拟的速度表示瞬时速度。

平均速度：我们定义平均速度为两个时间步t和r之间的位移（通过积分得到）除以时间间隔。从形式上讲，平均速度u可以表示为：

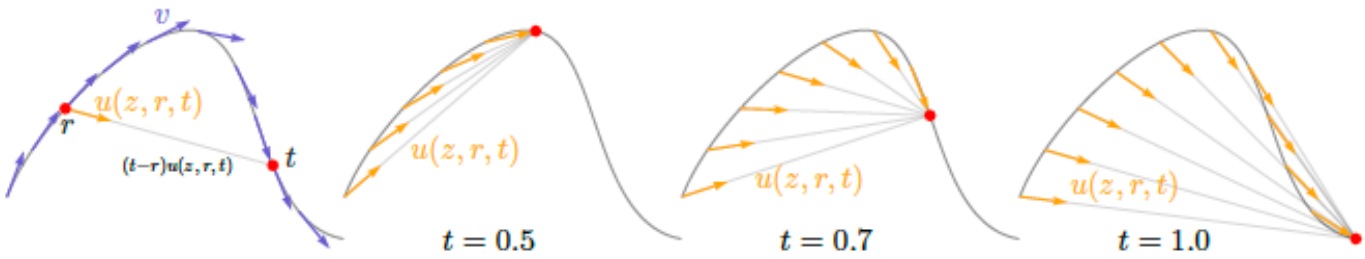
$$u(z_t, r, t) = \frac{1}{t-r}\int_r^t v(z_\tau, \tau)d\tau$$

为了强调概念上的差异，接下来我们用u表示平均速度，用v表示瞬时速度。$u(z_t, r, t)$是一个同时依赖r和t的场。一般情况下，平均速度u是瞬时速度v的泛函的结果，也就是说，$u = \mathcal{F}[v] = \frac{1}{t-r}\int_r^t v d\tau$。它是一个由$v$诱导得到的场，与神经网络无关。从概念上讲，正如瞬时速度v作为流匹配中的真值场一样，平均速度u在此处能够为神经网络训练提供一个潜在的真值场。

根据定义，u场满足一定的约束。随着$r \to t$，我们有$\lim_{r\to t} u = v$。此外，u自然满足“一致性”的形式：在$[r, t]$中采取一个大步的结果和在$[r, s]$和$[s, t]$中采取两个小步是一致的，即：

$$(t-r)u(z_t, r, t) = (s-r)u(z_s, r, s) + (t-s)u(z_t, s, t), \quad s \in [r, t]$$

这实际上就是$\int_r^t v d\tau = \int_r^s v d\tau + \int_s^t v d\tau$。因此，一个精确逼近真实$u$的网络自身就带有这种限制，不需要显式的约束条件。



上图为平均速度场$u(z, r, t)$。最左侧：瞬时速度v决定路径的切线方向，而平均速度$u(z, r, t)$一般不和v对齐。平均速度与位移$(t-r)u(z, r, t)$对齐。右边三个子图：t取三种不同的值时的平均速度。

Mean Flow的最终目标是使用神经网络$u_\theta(z_t, r, t)$来逼近平均速度。这样做的显著优点是，假设我们准确地近似了这个量，我们就可以用$u_\theta(\epsilon, 0, 1)$的单步估计来近似整个流。换句话说，该方法更适合于单步或少步生成，因为它不需要在推理时间显式地近似一个时间积分，而这是在建模瞬时速度v时所需要的。但是直接使用定义的平均速度作为真值场是难以训练的，因为它需要在训练过程中评估一个整体。关键见解是，平均速度的定义方程可以被重构以构造一个最终可用于训练的优化目标，即使只有瞬时速度是可获得的。

MeanFlow的性质：为了使得平均速度能够训练，我们首先重写定义方程：

$$(t-r)u(z_t, r, t) = \int_r^t v(z_\tau, \tau)d\tau$$

将r视为与t无关的变量，将两边同时对t求导，可得：

$$\frac{d}{dt}(t-r)u(z_t,r,t) = \frac{d}{dt}\int_r^t v(z_\tau,\tau)d\tau$$
$$\Updownarrow$$
$$u(z_t,r,t) + (t-r)\frac{d}{dt}u(z_t,r,t) = v(z_t,t)$$
$$\Updownarrow$$
$$\underbrace{u(z_t,r,t)}_{\text{average vel.}} = \underbrace{v(z_t,t)}_{\text{instant vel.}} - (t-r)\underbrace{\frac{d}{dt}u(z_t,r,t)}_{\text{time derivative}}$$

该方程就是“MeanFlow Identity”，它描述了u和v之间的关系。等号右侧为$u(z_t,r,t)$提供了一个“目标”形式，我们将利用它来构造一个损失函数来训练一个神经网络。为了有一个合适的目标，我们还必须进一步分解时间导数项，这就是接下来要讨论的问题。

计算关于时间的导数：为了计算方程中的$\frac{d}{dt}u$项，注意到$\frac{d}{dt}$表示总导数，可以用偏导数来展开：

$$\frac{d}{dt}u(z_t,r,t) = \frac{dz_t}{dt}\partial_z u + \frac{dr}{dt}\partial_r u + \frac{dt}{dt}\partial_t u$$

由于$\frac{dz_t}{dt} = v(z_t,t)$，$\frac{dr}{dt} = 0$，$\frac{dt}{dt} = 1$，我们可以得到$u$和$v$之间的另外一组关系：

$$\frac{d}{dt}u(z_t,r,t) = v(z_t,t)\partial_z u + \partial_t u$$

该式表明，总导数由$[\partial_z u, \partial_r u, \partial_t u]$（即函数$u(z_t,r,t)$的雅可比矩阵）和切向量$[v,0,1]$之间的雅可比向量积(JVP)给出。这可以通过`torch.func.jvp`得到。

利用平均速度进行训练：目前为止，这些公式和任何网络参数化无关。我们现在引入一个模型来学习u。形式上，我们将网络u_θ参数化，并鼓励其满足MeanFlow Identity。具体来说，我们就是要最小化下式：

$$\mathcal{L}(\theta) = \mathbb{E}\|u_\theta(z_t,r,t) - sg(u_{\text{tgt}})\|_2^2,$$
$$\text{where} \qquad u_{\text{tgt}} = v(z_t,t) - (t-r)(v(z_t,t)\partial_z u_\theta + \partial_t u_\theta)$$

u_{tgt}项就是有效的回归目标。该方程使用瞬时速度v作为唯一的真值信号，不需要进行积分计算。虽然目标函数应该包含u的导数，即∂u，但是它们被参数化的对应物，即∂u_θ替代。在损失函数中，按照通常的做法，在目标u_{tgt}中应该采用“Stop-Gradient”(sg)操作，即停止梯度。在此处，它通过雅可比向量积消除了双重反向传播的需要，从而避免了高阶计算与优化。尽管有这些可以优化的做法，但如果u_θ要实现零损失，很容易证明它会满足MeanFlow Identity，从而满足原始定义。

速度$v(z_t,t)$是流匹配中的边际速度。我们需要将其替换成条件速度，因此目标变成：

$$u_{\text{tgt}} = v_t - (t-r)(v_t\partial_z u_\theta + \partial_t u_\theta)$$

由于$v_t = a_t' x + b_t'\epsilon$是条件速度，并且默认$a_t = 1-t$，$b_t = t$，所以$v_t = \epsilon - x$。

总的来说，该方法在概念上很简单：它的行为类似于流匹配，关键的区别在于匹配目标后面加上了一项$-(t-r)(v_t\partial_z u_\theta + \partial_t u_\theta)$，这源于平均速度的引入。特别地，注意到如果限制到条件$t=r$，则第二项消失，该方法恰好与标准的Flow Matching匹配。

生成样本：使用MeanFlow模型进行采样，只需将时间积分替换为平均速度即可：

$$z_r = z_t - (t-r)u(z_t,r,t)$$

在一步采样的情况下，我们有$z_0 = z_1 - u(z_1,0,1)$，其中$z_1 = \epsilon \sim p_{\text{prior}}(\epsilon)$。虽然一步抽样是这项工作的主要重点，但实际上，给定这个方程，少步抽样也是卓有成效的。

Mean Flows with Guidance

该方法很显然支持无分类器指导(CFG)。我们将CFG视为底层真值场的一个属性，而不是在采样时刻简单地应用CFG(这会使得NFE翻倍)。这样的设置可以让我们既可以享受CFG带来的好处，也可以保持抽样过程中的1-NFE行为。

Ground-truth Fields：我们构建一个新的底层真值场v^{cfg}：

$$v^{\text{cfg}}(z_t, t|c) = \omega v(z_t, t|c) + (1-\omega)v(z_t, t)$$

这是有分类的条件场和无分类的条件场的线性组合，则：

$$v(z_t, t|x) = \mathbb{E}_{p_t(v_t|z_t,c)}[v_t] \quad \text{and} \quad v(z_t, t) = \mathbb{E}_c[v(z_t, t|c)]$$

上式中的v_t是条件速度。

遵循Mean Flow的精神，我们引入v^{cfg}对应的平均速度u^{cfg}。根据MeanFlow Identity，u^{cfg}满足：

$$u^{\text{cfg}}(z_t, r, t, |c) = v^{\text{cfg}}(z_t, t|c) - (t-r)\frac{d}{dt}u^{\text{cfg}}(z_t, r, t|c)$$

同样的，这里的v^{cfg}和u^{cfg}都是潜在的与神经网络无关的真值场。由于$v^{\text{cfg}}(z_t, t) = v(z_t, t)$，$v(z_t, t) = u(z_t, t, t)$，根据定义，这里的$v^{\text{cfg}}$可以重写成：

$$v^{\text{cfg}}(z_t, t|c) = \omega v(z_t, t|c) + (1-\omega)u^{\text{cfg}}(z_t, t, t)$$

Training with Guidance：通过上面两个公式，我们能够构建一个神经网络及其学习目标。我们直接用函数u_θ^{cfg}来参数化u^{cfg}。则我们可以得到目标：

$$\mathcal{L}(\theta) = \mathbb{E}\left\| u_\theta^{\text{cfg}}(z_t, r, t|c) - sg(u_{\text{tgt}}) \right\|_2^2,$$

$$\text{where} \quad u_{\text{tgt}} = \tilde{v}_t - (t-r)(\tilde{v}_t \partial_z u_\theta^{\text{cfg}} + \partial_t u_\theta^{\text{cfg}})$$

这个方程实际上跟前面定义的损失函数很像，唯一的不同就是v_t与$\tilde{v}_t$之间的区别：

$$\tilde{v}_t = \omega v_t + (1-\omega)u_\theta^{\text{cfg}}(z_t, t, t)$$

如果$\omega = 1$，那么这个损失函数就和前面的损失函数一模一样了。