

دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)  
دانشکده مهندسی کامپیوتر

## گزارش تحقیق درس یادگیری عمیق

# مدل‌های انتشاری در شبکه عصبی ترانسفورمر

نگارش

رضا آدینه پور

استاد درس

جناب آقای دکتر صفابخش

تیر ۱۴۰۳

## چکیده

چالش اصلی این تحقیق، بررسی و مقایسه کارایی ترنسفرمرها در برابر معماری‌های سنتی مانند U-Net در مدل‌های انتشار بوده است. نتایج به دست آمده نشان می‌دهند که ترنسفرمرها با مقیاس‌پذیری بهتر و کارایی بالاتر می‌توانند جایگزین مناسبی برای معماری‌های کنونی در مدل‌های انتشار باشند و به بهبود کیفیت تصاویر تولید شده کمک کنند. در این راستا، ترنسفرمرهای انتشار (DiTs) با افزایش عمق و عرض ترنسفرمرها و افزایش تعداد توکن‌های ورودی، توانسته‌اند بهبود قابل توجهی در معیار FID نشان دهن و نتایج برتری در بسته‌های محک ImageNet به دست آورند.

این تحقیق نشان می‌دهند که استفاده از ترنسفرمرها در مدل‌های انتشار می‌تواند راهکارهای نوینی برای بهبود کیفیت و کارایی در تولید تصاویر ارائه دهد. این رویکردها، علاوه بر بهبود عملکرد، می‌توانند به کاهش پیچیدگی‌های محاسباتی و افزایش سرعت فرآیندهای تولید تصویر کمک کنند. با توجه به این نتایج، ترنسفرمرها به عنوان یک جایگزین قوی برای معماری‌های سنتی در مدل‌های انتشار مطرح می‌شوند و می‌توانند به طور گسترده در کاربردهای مختلف مورد استفاده قرار گیرند.

کلیدواژه‌ها: یادگیری عمیق، مدل‌های انتشار، ترنسفرمر، یو-نت، کدگذار، کدگشا

# فهرست مطالب

۱	۱	مقدمه
۱	۱-۱	تعريف مسئله
۱	۱-۱-۱	انتشار چیست؟
۲	۱-۱-۲	مدل‌های انتشار در یادگیری ماشین چیستند؟
۳	۲-۱	اهمیت پژوهش
۴	۳-۱	ساختار پژوهش
۵	۲	مفاهیم اولیه
۵	۱-۲	مقدمه‌ای بر مدل‌ها انتشار
۶	۱-۱-۲	فرآیند انتشار پیشرو
۷	۱-۱-۲	فرایند انتشار معکوس
۱۰	۲-۲	شبکه عصبی کانولوشنی U-Net
۱۰	۱-۲-۲	ساختار شبکه
۱۱	۳-۲	ویژن ترنسفرم
۱۲	۱-۳-۲	نحوه عملکرد ویژن ترنسفرها
۱۳	۳	مقاله [۱]
۱۳	۱-۳	ایده اصلی مقاله
۱۳	۲-۳	کارهای پیشین

۱۴	۳-۳ مراحل انجام الگوریتم
۱۵	۴-۳ مزایا و معایب
۱۶	۵-۳ اجرای عملی الگوریتم
۱۶	۱-۵-۳ دانلود مخزن الگوریتم
۱۶	۲-۵-۳ نصب وابستگی‌ها پیشنهادها
۱۷	۶-۳ بیلد نرم‌افزار
۱۷	۷-۳ دانلود بنچ‌مارک‌ها
۱۸	۸-۳ لینک کردن بنچ‌مارک‌ها
۱۸	۹-۳ اجرا کردن بنچ‌مارک‌ها
۱۹	۴ مقایسه مقاله‌های [۱، ۲] و نتیجه‌گیری
۲۲	مراجع

# فهرست جداول

## فهرست تصاویر

۱-۱	ساختار مدل‌های مولد	۳
۲-۱	نحوه عملکرد مدل‌های DDPMs	۳
۲-۲	نمونه‌ای از آموزش مدل انتشار برای داده‌های ۲ بعدی [۴]	۸
۲-۲	ساختار شبکه U-Net	۱۱
۱-۳	مراحل انجام الگوریتم OpenPARF	۱۴
۱-۴	خروجی‌های زمانی الگوریتم DREAMPlace برای دو بنچ‌مارک مختلف	۲۰
۲-۴	خروجی‌های زمانی الگوریتم OpenPARF برای بنچ‌مارک ISPD ۲۰۱۷	۲۰

# فصل ۱

## مقدمه

هوش مصنوعی<sup>۱</sup> به طور مداوم در حال تکامل است تا مشکلات سخت و پیچیده را حل کند. تولید تصویر یکی از این مشکلات پیچیده برای مدل‌های هوش مصنوعی است. VAE<sup>۲</sup>ها، GAN<sup>۳</sup>ها و مدل‌های جریان عملکرد خوبی داشته‌اند اما در تولید تصاویر با وضوح بالا و دقت زیاد دچار مشکل بوده‌اند. از سوی دیگر، مدل‌های انتشار<sup>۴</sup> در تولید تصاویر با وضوح بالا و کیفیت متنوع با دقت بالا بسیار خوب عمل می‌کنند. در حال حاضر، آن‌ها در خط مقدم انقلاب هوش مصنوعی مولد (GenAI) قرار دارند که در همه جا دیده می‌شود. مدل‌هایی مانند DALL-E-3، GLIDE، OpenAI Imagen توسط گوگل، و Stable Diffusion از جمله مدل‌های انتشار پرطرفدار هستند. در ادامه به معرفی مسئله و برخی از پیش‌نیازها می‌پردازیم.

### ۱-۱ تعریف مسئله

#### ۱-۱-۱ انتشار چیست؟

انتشار یک پدیده طبیعی اساسی است که در سیستم‌های مختلف از جمله فیزیکی، شیمیایی و زیست‌شناسی مشاهده می‌شود.

این پدیده در زندگی روزمره به وضوح قابل مشاهده است. به عنوان مثال، در نظر بگیرید که عطر را اسپری می‌کنید. در ابتدا، مولکول‌های عطر به طور متراکم در نزدیکی نقطه اسپری قرار دارند. با گذشت زمان، مولکول‌ها

<sup>1</sup>Artificial Intelligence

<sup>2</sup>Generative Adversarial Networks

<sup>3</sup>Variational Autoencoder

<sup>4</sup>Diffusion Models

در محیط اطراف منتشر می‌شوند.

انتشار فرآیندی است که طی آن ذرات، اطلاعات یا انرژی از ناحیه‌ای با غلظت پایین‌تر حرکت می‌کنند. این اتفاق به این دلیل رخ می‌دهد که سیستم‌ها تمایل دارند به تعادل<sup>۵</sup> برسند، جایی که غلظت‌ها در سراسر سیستم یکسان می‌شود.

در یادگیری ماشین<sup>۶</sup> و تولید داده<sup>۷</sup>، انتشار به یک رویکرد خاص برای تولید داده‌ها با استفاده از یک فرآیند تصادفی مشابه با زنجیره مارکوف<sup>۸</sup> اشاره دارد. در این زمینه، مدل‌های انتشار نمونه‌های جدیدی از داده‌ها را با استفاده از داده‌های ساده‌تر ایجاد می‌کنند و به تدریج داده‌های پیچیده‌تر و واقعی‌تر تولید می‌شود.

## ۲-۱-۱ مدل‌های انتشار در یادگیری ماشین چیستند؟

مدل‌های انتشار مولد<sup>۹</sup> هستند، به این معنی که داده‌های جدیدی بر اساس داده‌هایی که بر روی آن‌ها آموزش دیده‌اند تولید می‌کنند. به عنوان مثال، یک مدل انتشار که بر روی مجموعه‌ای از داده‌های چهره‌های انسان آموزش دیده است، می‌تواند چهره‌های انسانی جدید و واقع‌گرایانه با ویژگی‌ها و حالت‌های مختلف تولید کند، حتی اگر آن چهره‌های خاص در مجموعه داده‌های اولیه وجود نداشته باشند.

برخلاف سایر مدل‌های مولدی مثل GAN، VAE و ... این مدل بر مدل‌سازی تکامل مرحله به مرحله توزیع داده‌ها از یک نقطه شروع ساده به یک توزیع پیچیده‌تر مرکز دارند. مفهوم اساسی مدل‌های انتشار این است که یک توزیع ساده مثل توزیع گاوی<sup>۱۰</sup>، را از طریق یک سری عملیات‌های قابل برگشت<sup>۱۱</sup> به یک توزیع داده پیچیده‌تر تبدیل کنند.

پس از اینکه مدل، فرآیند تبدیل را آموزش می‌بیند، می‌تواند نمونه‌های جدیدی را با شروع از یک نقطه در توزیع ساده و به تدریج "انتشار" آن به توزیع داده پیچیده مطلوب تولید کند.

در مدل‌های DDPMs<sup>۱۲</sup> با افروzen تدریجی نویز گاوی به داده‌های اصلی در فرآیند انتشار پیشرو<sup>۱۳</sup> و سپس یادگیری حذف نویز در فرآیند انتشار معکوس<sup>۱۴</sup> کار می‌کنند. «شکل ۲-۱»

<sup>5</sup>Equilibrium

<sup>6</sup>Machine Learning

<sup>7</sup>Data Generation

<sup>8</sup>Markov Chain

<sup>9</sup>Generative

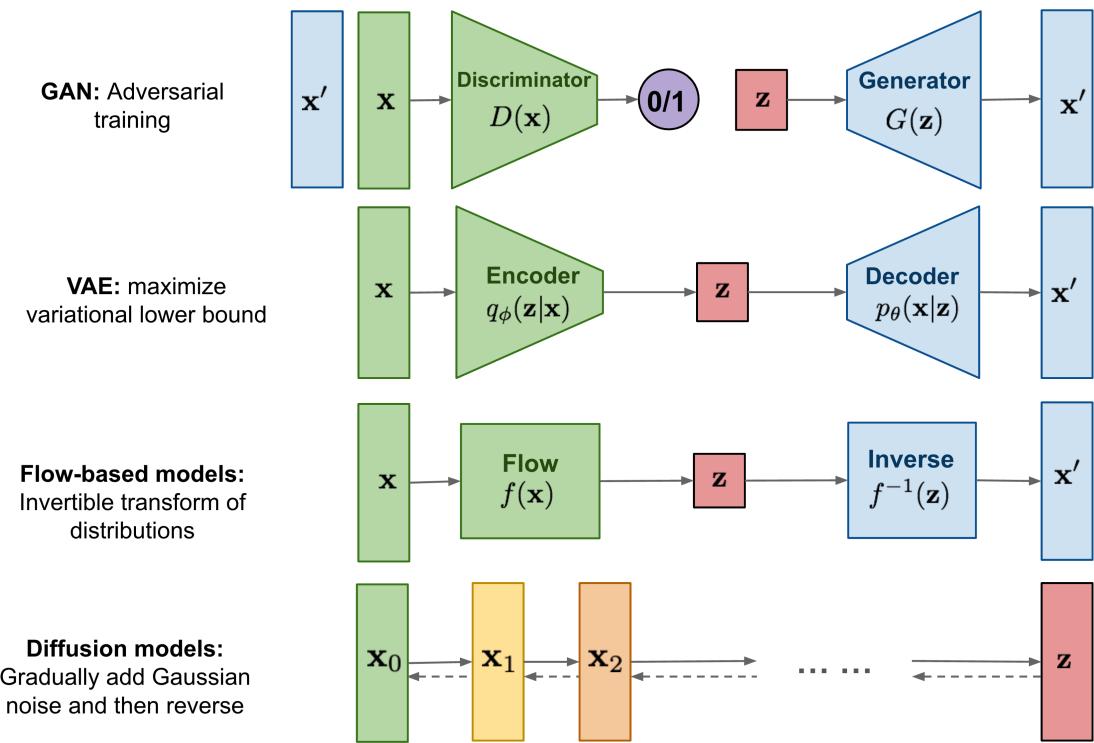
<sup>10</sup>Gaussian Distribution

<sup>11</sup>Invertible Operations

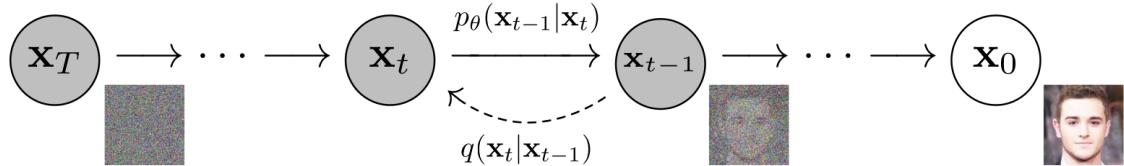
<sup>12</sup>Denoising Diffusion Probabilistic Models

<sup>13</sup>Forward Diffusion

<sup>14</sup>Reverse Diffusion



شکل ۱-۱: ساختار مدل‌های مولد



شکل ۲-۱: نحوه عملکرد مدل‌های DDPMs [۵]

## ۲-۱ اهمیت پژوهش

مدل‌های انتشار با شبیه‌سازی فرآیندهای تصادفی قادر به تولید نمونه‌های واقعی‌تری هستند که در کاربردهای مختلفی مانند ترمیم تصاویر، نویززدایی و تولید تصاویر با وضوح بالا استفاده می‌شوند. به ویژه، ترانسفورمرهای انتشار (DiT) به عنوان یک نوآوری در این حوزه با استفاده از معماری ترانسفورمر، قدرت بیشتری در مدل‌سازی پیچیدگی‌های داده‌ها دارند. DiT‌ها با توانایی مقیاس‌پذیری بالا و کاهش مداوم معیار FID، بهبود قابل توجهی در کیفیت و واقع‌گرایی نمونه‌های تولیدی ارائه می‌دهند. این ویژگی‌ها DiT‌ها را به ابزار قدرتمندی برای وظایف پیچیده‌تری مانند سنتز تصویر و بهبود کیفیت تصویر تبدیل می‌کنند.

### ۳-۱ ساختار پژوهش

این پژوهش در ۵ فصل انجام شده است. در فصل ۱ به مقدمه و اهمیت موضوع پژوهش پرداخته شده است. در فصل ۲ به مفاهیم اولیه و پیش‌نیازها پرداخته شده است. در ادامه در فصل سوم پژوهش به بررسی کارهای پیشین انجام شده در این زمینه پرداخت شده است. در فصل چهارم به بررسی دقیق و جزئی مقالات مطالعه شده در این پژوهش پرداخته شده است و در فصل پایانی، جمع‌بندی و نتیجه‌گیری پژوهش ارائه شده است.

## فصل ۲

### مفاهیم اولیه

همانطور که در فصل ۱ بیان شد، برای درک بهتر مدل‌های انتشار، ابتدا می‌بایست پیش‌نیازها و مفاهیم اولیه این زمینه را بیان کنیم. در ادامه به معرفی و بررسی مدل‌های انتشار و ساختارهای معروف و مورد استفاده در مقالات مرتبط می‌پردازیم:

#### ۱-۲ مقدمه ای بر مدل‌ها انتشار

مدل‌های انتشار نوعی مدل مولد هستند که یک زنجیره مارکوف را شبیه‌سازی می‌کنند تا از یک توزیع اولیه ساده به توزیع داده‌ها پیچیده انتقال یابند. این فرآیند شبیه به حرکت پراونی یک ذره است که هر مرحله آن یک حرکت تصادفی کوچک است. به همین دلیل به آن‌ها "مدل‌های انتشار" گفته می‌شود.

مدل‌های انتشار در کاربردهای مختلفی مانند نویز‌زدایی<sup>۱</sup>، افزایش وضوح تصویر<sup>۲</sup> و ترمیم تصاویر<sup>۳</sup> استفاده می‌شود. یکی از مزایای کلیدی مدل‌های انتشار توانایی آن‌ها در تولید نمونه‌های با کیفیت بالا است که آن‌ها را به‌ویژه برای وظایفی مانند سنتر تصویر بسیار مفید می‌سازد.

به‌طور کلی فرآیند انتشار از دو مرحله تشکیل می‌شود:

۱. انتشار پیش‌رو

۲. انتشار پس‌رو

<sup>1</sup>Denoising

<sup>2</sup>Super-resolution

<sup>3</sup>Inpainting

که در ادامه به توضیح هر کدام می‌پردازیم:

## ۱-۱-۲ فرآیند انتشار پیشرو

فرآیند انتشار پیشرو یک زنگیره مارکوف از مراحل انتشار است که در آن به تدریج و به صورت تصادفی نویز به داده‌های اصلی اضافه می‌کنیم.

با فرض اینکه یک نقطه داده از یک توزیع داده واقعی  $(x \sim q)$  نمونه‌برداری شده باشد، یک فرآیند انتشار پیشرو را تعریف کنیم که در آن مقدار کمی نویز گاووسی به نمونه در  $T$  مرحله اضافه می‌کنیم، به طوری که یک دنباله از نمونه‌های نویزی  $x_1, \dots, x_T$  تولید می‌شود. اندازه مراحل با یک برنامه تغییرپذیری  $\{\beta_t \in (0, 1)\}_{t=1}^T$  کنترل می‌شود.

$$q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t \mathbf{I}) \quad (1-2)$$

$$q(x_{1:T} | x_0) = \prod_{t=1}^T q(x_t | x_{t-1}) \quad (2-2)$$

با بزرگتر شدن  $t$ ، نمونه داده  $x$  به تدریج ویژگی‌های قابل تشخیص خود را از دست می‌دهد در نهایت، زمانی که  $T \rightarrow \infty$  معادل یک توزیع گاووسی همسانگرد<sup>۴</sup> خواهد بود.

یکی از ویژگی‌های جالب فرآیند فوق این است که می‌توانیم  $x_t$  را در هر لحظه زمانی دلخواه  $t$  با استفاده از ترفند بازپارامتری‌سازی<sup>۵</sup> به صورت بسته نمونه‌برداری کنیم. اگر  $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$  و  $\bar{\beta}_t = 1 - \beta_t$  باشد داریم:

<sup>4</sup>Isotropic

<sup>5</sup>Reparameterization

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \tilde{\boldsymbol{\epsilon}}_{t-2} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}
\end{aligned} \tag{3-2}$$

$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$

توجه به این نکته مهم است که وقتی دو توزیع گاوی با واریانس‌های مختلف  $\mathcal{N}(\mathbf{0}, \sigma_1^2 \mathbf{I})$  و  $\mathcal{N}(\mathbf{0}, \sigma_2^2 \mathbf{I})$  را باهم ترکیب می‌کنیم، توزیع جدید  $\mathcal{N}(\mathbf{0}, (\sigma_1^2 + \sigma_2^2) \mathbf{I})$  است. در اینجا انحراف استاندارد ترکیبی برابر است با:

$$\sqrt{(1 - \alpha_t) + \alpha_t(1 - \alpha_{t-1})} = \sqrt{1 - \alpha_t \alpha_{t-1}} \tag{4-2}$$

معمولًاً، می‌توانیم گام به روزرسانی بزرگتری داشته باشیم وقتی که نمونه‌ها نویز بیشتری پیدا می‌کنند، بنابراین  $\cdot \bar{\alpha}_T < \beta_1 < \beta_2 < \dots < \beta_T$  و در نتیجه

## ۴-۱-۲ فرایند انتشار معکوس

فرایند انتشار معکوس سعی می‌کند فرایند انتشار پیشرو را به صورت معکوس انجام دهد و از داده‌های نویزی، داده‌های اصلی را با کیفیت بالا تولید کند.

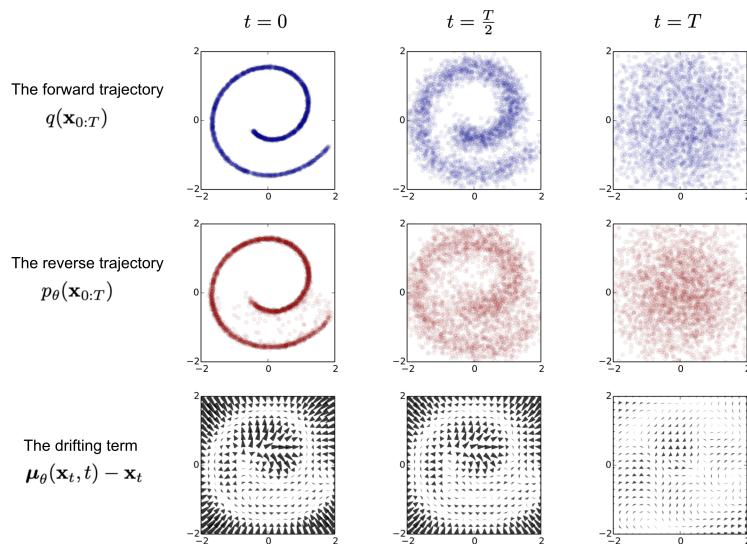
اگر بتوانیم فرایند فوق را معکوس کنیم و از  $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$  نمونه‌برداری کنیم، قادر خواهیم بود نمونه واقعی را از یک ورودی نویز گاوی بازسازی کنیم:

$$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{5-2}$$

اگر مقدار  $\beta_t$  به اندازه کافی کوچک باشد،  $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$  نیز گاوی خواهد بود. متاسفانه، نمی‌توانیم به راحتی  $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$  را تخمین بزنیم زیرا نیاز به استفاده از کل مجموعه داده‌ها دارد و بنابراین باید یک مدل  $p_\theta$  را بیاموزیم تا این احتمالات شرطی را تقریب بزند تا بتوانیم فرایند انتشار معکوس را اجرا کنیم.

$$p_{\theta}(\mathbf{x}_{\circ:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) \quad (8-2)$$

$$p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t)) \quad (8-3)$$



شکل ۱-۲: نمونه‌ای از آموزش مدل انتشار برای داده‌های ۲ بعدی [۴]

قابل توجه است که احتمال شرطی معکوس زمانی که بر روی  $\mathbf{x}_\circ$  شرطی شود، به صورت زیر قابل محاسبه است:

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_\circ) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_\circ), \tilde{\beta}_t \mathbf{I}) \quad (8-4)$$

با استفاده از قاعده بیز<sup>۶</sup>، می‌توان نوشت:

---

<sup>6</sup>Bayes' rule

$$\begin{aligned}
q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_o) &= \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_o) q(\mathbf{x}_{t-1} | \mathbf{x}_o)}{q(\mathbf{x}_t | \mathbf{x}_o)} \\
&\propto \exp \left( -\frac{1}{2} \left( \frac{(\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_{t-1})^T}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_t} \mathbf{x}_o)^T}{1 - \alpha_{t-1}^-} \right. \right. \\
&\quad \left. \left. - \frac{(\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_o)^T}{1 - \bar{\alpha}_t} \right) \right) \\
&= \exp \left( -\frac{1}{2} \left( \frac{\mathbf{x}_t^T - 2\sqrt{\alpha_t} \mathbf{x}_t \mathbf{x}_{t-1} + \alpha_t \mathbf{x}_{t-1}^T}{\beta_t} \right. \right. \\
&\quad \left. \left. + \frac{\mathbf{x}_{t-1}^T - 2\sqrt{\bar{\alpha}_t} \mathbf{x}_{t-1} \mathbf{x}_o + \bar{\alpha}_t \mathbf{x}_o^T}{1 - \alpha_{t-1}^-} \right. \right. \\
&\quad \left. \left. - \frac{\mathbf{x}_t^T - 2\sqrt{\alpha_t} \mathbf{x}_t \mathbf{x}_o + \bar{\alpha}_t \mathbf{x}_o^T}{1 - \bar{\alpha}_t} \right) \right) \\
&= \exp \left( -\frac{1}{2} \left( \left( \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \alpha_{t-1}^-} \right) \mathbf{x}_{t-1}^T \right. \right. \\
&\quad \left. \left. - \left( \frac{2\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_t}}{1 - \alpha_{t-1}^-} \mathbf{x}_o \right) \mathbf{x}_{t-1}^T \right. \right. \\
&\quad \left. \left. + \frac{2\sqrt{\bar{\alpha}_t}}{1 - \bar{\alpha}_t} \mathbf{x}_t + C(\mathbf{x}_t, \mathbf{x}_o) \right) \right) \\
\end{aligned} \tag{9-2}$$

با پیروی از تابع چگالی گاوی استاندارد<sup>۴</sup>، میانگین و واریانس به صورت زیر بیان می‌شوند. (به یاد داریم که  $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$  و  $\alpha_t = 1 - \beta_t$ )

$$\tilde{\beta}_t = \left( \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \alpha_{t-1}^-} \right)^{-1} = \left( \frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1 - \alpha_{t-1}^-)} \right)^{-1} = \beta_t \frac{1 - \alpha_{t-1}^-}{1 - \bar{\alpha}_t} \tag{10-2}$$

$$\begin{aligned}
\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_o) &= \left( \frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\alpha_{t-1}^-}}{1 - \alpha_{t-1}^-} \mathbf{x}_o \right) \left( \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \alpha_{t-1}^-} \right)^{-1} \\
&= (\sqrt{\alpha_t} \mathbf{x}_t + \sqrt{\bar{\alpha}_t} \mathbf{x}_o) \left( \frac{1 - \alpha_{t-1}^-}{1 - \bar{\alpha}_t} \right) \\
&= \frac{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_t \beta_t}}{1 - \bar{\alpha}_t} \mathbf{x}_o.
\end{aligned} \tag{11-2}$$

---

<sup>۴</sup>Standard Gaussian Density Function

## ۲-۲ شبکه عصبی کانولوشنی U-Net

در معماری‌های رایج شبکه‌های کانولوشنی، کانال‌های ویژگی به منظور استخراج اطلاعات معنایی به طور متناوب از لایه‌های ادغام<sup>۸</sup> عبور می‌کنند. لایه‌های ادغام با کاهش اطلاعات فضایی میدان دید لایه‌های بعدی را افزایش می‌دهند و از این طریق شبکه می‌تواند اطلاعات معنایی را از تصاویر استخراج کند. هرچه از ورودی یک شبکه کانولوشنی به سمت خروجی آن حرکت کنیم، اطلاعات فضایی کاسته شده و بر اطلاعات معنایی افزوده می‌شود؛ بنابراین به نظر می‌رسد بین اطلاعات فضایی و معنایی در این شبکه‌ها مصالحه<sup>۹</sup> ای وجود دارد. که افزایش هریک، موجب کاهش دیگری می‌شود. شبکه یو-نت به این چالش بزرگ از طریق معماری خاص خود و اتصالات پرشی رسیدگی می‌کند. این شبکه دارای یک ساختار متقارن-کدگذار-کدگشا می‌باشد (شکل ۲-۲) که در آن کانال‌های ویژگی در سطوح مختلفی از بخش کدگذار، از طریق اتصال پرشی، به کانال‌های ویژگی کدگشا الحق<sup>۱۰</sup> می‌شود. این ساختار برخلاف ساختار شبکه تمام کانولوشنی [۶]، که خروجی نهایی را در یک مرحله نمونه افزایی<sup>۱۱</sup> ایجاد می‌کند، باعث می‌شود بازیابی اطلاعات فضایی بهتر صورت بگیرد. تفاوت دیگر این دو شبکه این است که اتصالات پرشی در شبکه یو-نت به جای استفاده از عمل جمع، از عمل الحق استفاده می‌کند. این ساختار متقارن، به همراه تکنیک‌های افرونگی داده، باعث می‌شود شبکه یو-نت بتواند علاوه بر کسب دقت بالا، از تعداد بسیار کمی داده‌های آموزشی را یاد بگیرد.

### ۱-۲-۲ ساختار شبکه

شکل (۲-۲) ساختار شبکه یو-نت را نشان می‌دهد. این شبکه از یک بخش کدگذار «سمت چپ» و یک بخش کدگشا «سمت راست» تشکیل شده است. هر مرحله از بخش کدگذار، شامل دو لایه کانولوشنی  $3 \times 3$  با یک تابع فعالسازی<sup>۱۲</sup> ReLU در هر کدام و یک لایه ادغام حداقل<sup>۱۳</sup> با سایز  $2 \times 2$  و گام ۲ می‌باشد. در هر مرحله از بخش کدگشا، ابتدا یک لایه نمونه افزایی به همراه یک لایه کانولوشنی  $2 \times 2$  «کانولوشن افزاینده<sup>۱۴</sup>» تعداد کانال‌های ویژگی را نصف و ابعاد آن را دو برابر می‌کند. سپس کانال‌های ویژگی با کانال‌های برش داده شده متناظر از بخش کدگذار الحق می‌شوند. در ادامه دولایه کانولوشنی، مشابه آنچه در بخش کدگذار وجود دارد، روی کانال‌های ویژگی اعمال می‌شود. در لایه‌های کانولوشنی این شبکه عمل گسترش مرز با صفر<sup>۱۵</sup> انجام نمی‌شود و بنابراین برای الحق کانال‌های ویژگی، همانطور که اشاره شد، کانال‌های ویژگی بخش کدگذار بریده می‌شوند.

<sup>8</sup>Pooling

<sup>9</sup>Trade-off

<sup>10</sup>Concatenate

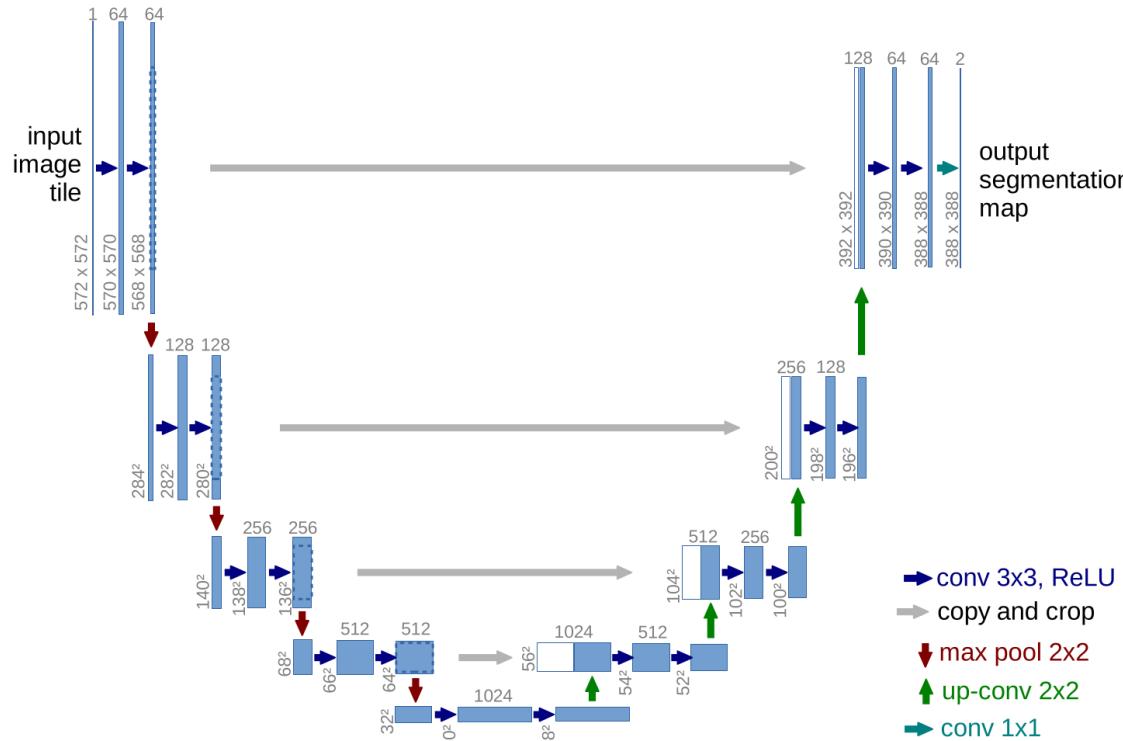
<sup>11</sup>Up-Sampling

<sup>12</sup>Activation Function

<sup>13</sup>Max Pooling

<sup>14</sup>Up-Convolution

<sup>15</sup>Zero Padding



شکل ۲-۲: ساختار شبکه [5] U-Net

در انتهای بخش کدگشا یک لایه کانولوشنی  $1 \times 1$  تعداد کانال‌های ویژگی «۶۴» را به تعداد کلاس‌های مسئله تبدیل می‌کند.

### ۳-۲ ویژن ترنسفورمر

ویژن ترنسفورمرها (ViT) مدلی جدید در زمینه بینایی کامپیوترا<sup>۱۶</sup> هستند که مدل‌های ترنسفورمر را که در اصل برای وظایف پردازش زبان طبیعی طراحی شده‌اند، به وظایف طبقه‌بندی تصاویر اعمال می‌کنند. برخلاف شبکه‌های عصبی کانولوشنی سنتی که تصاویر را به صورت سلسله مراتبی پردازش می‌کنند، ViT ها به صورت موازی تصاویر را پردازش می‌کنند.

شبکه‌های ViT تصاویر را به عنوان دنباله‌ای از تکه‌ها در نظر می‌گیرند و وابستگی‌های جهانی<sup>۱۷</sup> بین آن‌ها را ثبت می‌کنند. این ویژگی به آن‌ها امکان مدل‌سازی تعاملات پیکسلی با برد بلند را می‌دهد. یکی از مزایای کلیدی ViT ها قابلیت مقیاس‌پذیری آن‌ها است. آن‌ها می‌توانند بر روی مجموعه داده‌های بزرگ آموزش بینند و تصاویر بزرگ را به عنوان ورودی قبول کنند.

<sup>16</sup>Computer Vision

<sup>17</sup>Global Dependencies

## ۴-۲ نحوه عملکرد ویژن ترنسفرها

### ۱-۴-۲ بینیان ترنسفرمر

برای درک چگونگی عملکرد ویژن ترنسفرمرها، لازم است مفاهیم پایه‌ای معماری ترنسفرمر مانند توجه به خود<sup>۱۸</sup> را درک کنیم. توجه به خود یک مکانیزم است که به مدل اجازه می‌دهد تا هنگام پیش‌بینی، اهمیت عناصر مختلف در یک دنباله را وزن‌دهی کند و در نتیجه به نتایج چشمگیری در وظایف مبتنی بر دنباله دست یابد.

---

<sup>18</sup>Self-Attention

# فصل ۳

## مقاله [۱]

### ۱-۳ ایده اصلی مقاله

ایده اصلی این مقاله نیز، حول محور مسئله مسیریابی در طراحی‌ها با استفاده از تکنیک‌های هوش مصنوعی قرار دارد با این تفاوت که در این مقاله مسیریابی‌ها مختص FPGA است و الگوریتم پیشنهاد شده برای مسیریابی بهتر درون FPGA پیشنهاد شده است. الگوریتم ارائه شده در این مقاله نیز متن‌باز است و مبتنی بر Python و کتابخانه PyTorch است و از موازی سازی در سطح GPU پشتیبانی می‌کند. در این مقاله ادعا شده است که طول مسیرها تا ۴۵٪ تا ۷٪ کوتاه شده و سرعت مسیریابی نیز بیشتر از ۲ برابر روش‌های معمولی شده است.

### ۲-۳ کارهای پیشین

همانند مقاله قبل در این مقاله نیز اشاره شده است که روش‌های قدیمی که به روش‌های CAD<sup>۱</sup> معروف هستند در گذشته به طور عمده مورد استفاده قرار می‌گرفته است و مرحله مسیریابی به تنها ۴۱ تا ۸۶ درصد از کل زمان‌بندی مسیر طراحی CAD را به خود اختصاص می‌دهد [۱] و از این رو کاری بسیار زمانبر است. بنابر این کیفیت و کارایی الگوریتم‌های PAR<sup>۲</sup> بر طراحی بهینه بسیار تاثیرگذار هستند.

الگوریتم‌های بسیار زیادی مطرح شده است که همگی آنها مبتنی بر ابزار CAD ارائه شده توسط شرکت‌های سازنده FPGA هستند.

Computer-Aided Design<sup>۱</sup>  
Placement and Routing<sup>۲</sup>

برای اینکه شرکت های تولید کننده FPGA از تولید مجدد ابزار CAD به خصوص، برای یک خاص جلوگیری کنند، ابزاری متن باز مبتنی بر کتابخانه یادگیری عمیق PyTorch ارائه شده است.

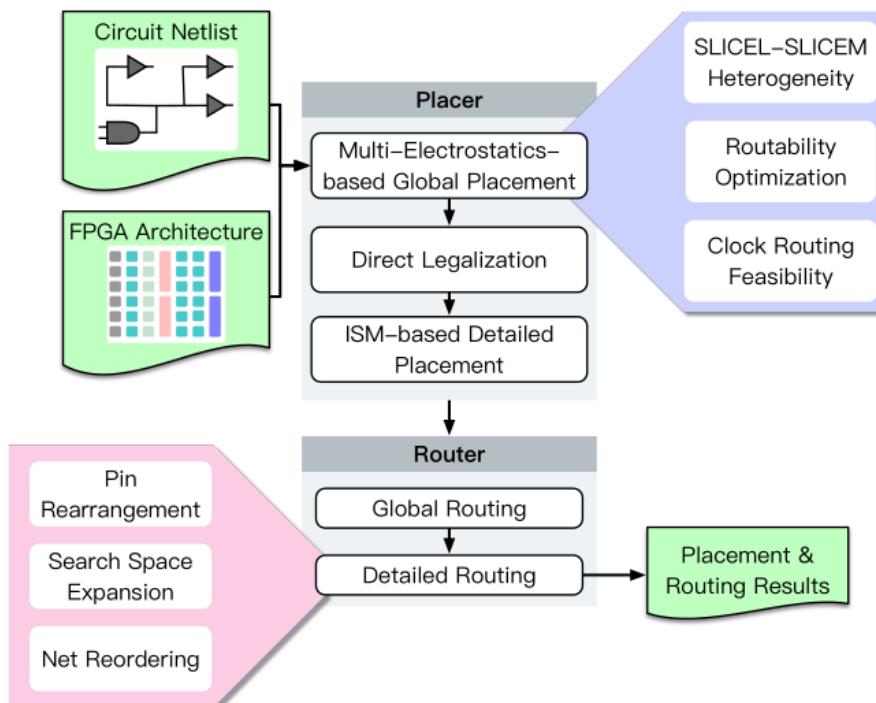
ایده های اصلی این مقاله را می توان به صورت زیر دسته بندی کرد:

- ابزار ارائه شده به نام OpenPARF است که ابزاری متن باز برای مسیریابی FPGA های پیشرفته، مبتنی بر یادگیری عمیق و قابل اجرا بر روی دو پلتفرم CPU و GPU است.

• این ابزار الگوریتم های مسیریابی غیر خطی را بر اساس یک میدان الکترواستاتیکی نا متقارن پیاده سازی می کند که قادر است به نتایج مکانیابی برتر تحت محدودیت های مختلف مسیریابی دست پیدا کند.

### ۳-۳ مراحل انجام الگوریتم

مراحل انجام این الگوریتم در «شکل ۱-۳» نشان داده شده است. همانطور مشخص است، این الگوریتم دارای ۲ مرحله چینش و مسیریابی است.



شکل ۱-۳: مراحل انجام الگوریتم

در ابتدا فایل های اطلاعات مکان ها «فایل هایی با فرمت bookshelf» و فایل های ساختار و معماری مسیریابی مورد نظر «فایل هایی با فرمت XML» در ساختار داده داخلی OpenPARF ساخته می شود و

در مرحله دوم، بر اساس ساختار داده داخلی، OpenPARF از الگوریتم های مکانیابی مبتنی بر میدان های الکترواستاتیکی چندگانه مسئله مسیریابی را در یک چارچوب بهینه سازی استاندارد استفاده می کند.

پس از انجام مسیریابی در مرحله اول، الگوریتم تلاش می کند با انجام مسیریابی های دقیق تر و با جزئیات بیشتر، مشکلات موجود در مسیریابی اولیه انجام شده را بر طرف نماید.

پس این الگوریتم مسیریابی از دو مرحله تشکیل می شود:

- مسیریابی اصلی سطح دانه درشت

- مسیریابی دقیق و جزئی سطح دانه ریز

در مرحله اول یک مسیر اصلی و کلی پیشنهاد می شود و در مرحله دوم، مسیریابی به صورت دقیق و منطقه به منطقه بررسی می شود و بهترین خروجی را تولید می کند.

## ۴-۳ مزایا و معایب

از مزایای روش پیشنهاد شده می توان به موارد زیر اشاره کرد:

- این مقاله نیز همانند قبلی یکی از مزیت های آن متن باز بودن آن است
- توسعه الگوریتم به دو زبان Python و C++
- توسعه الگوریتم بر دو بستر CPU و GPU
- ارائه نتایج برای دو بنچمارک صنعتی و اکادمیک جدید
- مقایسه بسیار گسترده LUT ها و FF ها و سایر منابع مصرفی FPGA به هنگام استفاده از این الگوریتم
- ... و ...

در کنار بیان مزایا می بایست به معایب پژوهه را هم بیان کرد. در ادامه چند مورد از معایب پژوهه انجام شده را بیان می کنیم:

- عدم اشاره مستقیم مقاله به ساختار شبکه عصبی استفاده شده در این الگوریتم

- نیازمند بودن به سیستمی قوی برای اجرای این الگوریتم. چرا که این الگوریتم بر روی سیستمی بسیار قوی آموزش داده شده است و به این نکته اشاره نشده است که مینیمم امکانات سیستم مورد نظر چه باید باشد که بتوان این الگوریتم را بر روی آن اجرا کرد.
- نیازمند بودن الگوریتم به FPGA های پیشرفته با منابع داخلی بالا. چرا که به نظر بnde این الگوریتم بر روی FPGA های قدیمی با منابع داخلی محدود قابل اجرا نمی باشد
- عدم صحبت از توان مصرفی الگوریتم
- عدم ارائه گزارش از خطاهای ناشی از اجرای الگوریتم
- ... و ...

## ۵-۳ اجرای عملی الگوریتم

### ۱-۵-۳ دانلود مخزن الگوریتم

با استفاده از دستور زیر، مخزن<sup>۳</sup> الگوریتم را دانلود می کنیم.

```
$ git clone https://github.com/PKU-IDEA/OpenPARF.git
```

### ۲-۵-۳ نصب وابستگی ها پیشنيازها

وابستگی های<sup>۴</sup> مورد نیاز را به صورت زیر نصب می کنیم:

```
# * create and activate conda environment
$ mamba create --name openparf python=3.7
$ mamba activate openparf
# * common packages
$ mamba install cmake boost bison
# * Pytorch 1.7.1. Other version may also work, but have not been tested.
$ mamba install pytorch==1.7.1 torchvision==0.8.2 cudatoolkit=11.0 -c pytorch
```

---

Repository<sup>۳</sup>  
Dependencies<sup>۴</sup>

```
# * python packages  
$ pip install hummingbird-ml pyyaml networkx tqdm
```

## ۶-۳ بیلد نرم افزار

به صورت زیر، OpenPARF را بیلد می کنیم:

```
$ mkdir build  
$ cd build  
$ cmake ..../OpenPARF -DCMAKE_PREFIX_PATH=$CONDA_PREFIX  
-DPYTHON_EXECUTABLE=$(which python)-DPython3_EXECUTABLE=$(which python)  
-DCMAKE_INSTALL_PREFIX=<installation directory>  
$ make -j8  
$ make install
```

## ۷-۳ دانلود بنچ مارک ها

بنچ مارک های مورد استفاده در این مقاله را می توان از آدرس های زیر دانلود نمود:

1. ISPD 2016 FPGA Placement Benchmarks [[download](#)]
2. ISPD 2016 FPGA Placement Flexshelf Benchmarks [[download](#)]
3. ISPD 2017 FPGA Placement Benchmarks [[download](#)]
4. ISPD 2017 FPGA Placement Flexshelf Benchmarks [[download](#)]

می توان به صورت زیر نیز بنچ مارک ها را در ترمینال دانلود کرد:

```
$ curl <url> --output <output filename>
```

## ۸-۳ لینک کردن بنچمارک ها

با استفاده از دستورات زیر، می‌توان بنچمارک‌های دانلود شده را لینک کرد:

```
$ ln -s <benchmark directory>/ispd2016 <installation  
directory>/benchmarks/ispd2016  
  
$ ln -s <benchmark directory>/ispd2016_flexport <installation  
directory>/benchmarks/ispd2016_flexport  
  
$ ln -s <benchmark directory>/ispd2017 <installation  
directory>/benchmarks/ispd2017  
  
$ ln -s <benchmark directory>/ispd2017_flexport <installation  
directory>/benchmarks/ispd2017_flexport
```

## ۹-۳ اجرا کردن بنچمارک ها

بنچمارک‌ها را به صورت زیر می‌توان اجرا<sup>۵</sup> نمود:

```
$ cd <installation directory>  
$ python openparf.py --config unittest/regression/ispd2016_flexport/  
FPGA01_flexport.json
```

## فصل ۴

### مقایسه مقاله های [۱، ۲] و نتیجه گیری

هر دو مقاله بررسی شده، به ارائه ساختاری متن باز جهت انجام دو فرایند مهم در طراحی به نام های چینش و مسیریابی با استفاده از تکنیک های یادگیری عمیق پرداخته است. با این تفاوت که در مقاله [۲] محدودیتی بر روی مدار یا ساختار مورد بخث گذاشته نشده است و می‌توان الگوریتم را برای هر مدار فشرده و خیلی فشرده «VLSI» ای آموزش داد و از آن در فرایند طراحی استفاده نمود. اما در مقاله [۱] الگوریتم ارائه شده فقط مخصوص چینش و مسیریابی FPGA هایی با منابع داخلی فراوان است. از این رو، کاربرد و عمومیت مقاله [۲] نسبت به مقاله [۱] بیشتر است.

هر دو مقاله برای انجام فرایند آموزش از سخت‌افزار های پیشرفته و گران قیمتی استفاده کرده‌اند.

در [۲] از سروری ۴۰ هسته لینوکسی با سی‌پی‌یو Intel E5-۲۶۹۸ با فرکانس کاری ۲/۲ گیگاهرتز و کارت‌گرافیک NVIDIA Tesla V۱۰۰ برای فرایند آموزش استفاده شده است.

و در [۱] از سروری لینوکس بیس که CPU آن Intel Xeon ۶۲۳۰ با فرکانس کاری ۱۰/۲ گیگاهرتز و ۴۰ هسته هسته، با کارت‌گرافیک NVIDIA RTX ۲۰۸۰ Ti و حافظه رم ۵۱۲ گیگابایتی استفاده شده است.

در هر دو مقاله سیستم‌های استفاده شده، سیستم‌های قوی و خاص منظوره ای هستند که کمتر در دسترس عموم مردم است. به همین دلیل ممکن است نتوان این الگوریتم‌ها را با سیستم‌های معمولی توسعه داد و این شاید به نوعی یکی از عیوب‌های این دو الگوریتم به حساب آید.

در [۲] الگوریتم با دو بنچ‌کارک اکادمیک و صنعتی تست شده است که خروجی‌های آن به صورت زیر است:  
«شکل ۱-۴»

در [۱] نیز خروجی به ازای بنچ‌کارک ISPD ۲۰۱۷ مشاهده شده است. «شکل ۲-۴» همانطور که در «شکل ۱-۴» مشاهده می‌شود، در الگوریتم DREAMPlace با افزایش تعداد سلول‌های

Table 2: Experimental results on ISPD 2005 benchmarks [25].

Design	#cells	#nets	RePIAe [6] (40 Threads)					DREAMPlace					
			HPWL	GP (s)	LG (s)	DP (s)	Total (s)	HPWL	GP (s)	LG (s)	DP (s)	IO (s)	Total (s)
adaptec1	211K	221K	73.26	106	5	24	139	73.30	5	0.5	25	5	37
adaptec2	255K	266K	81.87	194	7	30	236	82.19	6	0.5	32	6	47
adaptec3	452K	467K	193.20	382	22	53	466	194.12	10	1	58	10	82
adaptec4	496K	516K	175.23	434	21	61	524	174.43	11	2	65	11	92
bigblue1	278K	284K	89.85	168	3	30	206	89.43	7	0.3	35	6	51
bigblue2	558K	577K	138.09	383	22	90	505	136.69	11	9	90	12	125
bigblue3	1097K	1123K	304.83	967	46	138	1171	303.99	22	3	145	24	198
bigblue4	2177K	2230K	743.73	2037	56	329	2463	743.75	43	9	336	54	446
ratio	-	-	1.002	34.8	10.6	0.9	5.0	1.000	1.0	1.0	1.0	-	1.0

Table 3: Experimental results on industrial benchmarks.

Design	#cells	#nets	RePIAe [6] (40 Threads)					DREAMPlace					
			HPWL	GP (s)	LG (s)	DP (s)	Total (s)	HPWL	GP (s)	LG (s)	DP (s)	IO (s)	Total (s)
design1	1345K	1389K	340.42	974	46	155	1216	340.87	24	4	179	34	244
design2	1306K	1355K	275.46	1001	44	152	1238	275.76	24	5	180	32	244
design3	2265K	2276K	524.35	1767	84	257	2189	522.79	37	14	305	55	414
design4	1525K	1528K	455.22	1130	55	187	1424	454.38	26	8	207	37	281
design5	1316K	1364K	287.24	955	46	153	1193	288.41	22	4	186	33	248
design6	10504K	10747K	NA	>9100	NA	NA	NA	2356.88	282	73	1681	276	2323
ratio	-	-	1.000	43.1	9.5	0.9	5.0	1.000	1.0	1.0	1.0	-	1.0

شکل ۱-۴: خروجی های زمانی الگوریتم DREAMPlace برای دو بنچمارک مختلف

Design	#LUT/#FF/#BRAM/#DSP	#Clock	RippleFPGA [13]			OpenPARF		
			PRT	RRT	RWL	PRT	RRT	RWL
CLK-FPGA01	211K/324K/164/75	32	278	10	238.54	131	10	205.44
CLK-FPGA02	230K/280K/236/112	35	250	15	261.85	127	14	246.65
CLK-FPGA03	410K/481K/850/395	57	537	24	648.69	206	24	594.00
CLK-FPGA04	309K/372K/467/224	44	346	18	440.09	157	19	420.30
CLK-FPGA05	393K/469K/798/150	56	501	25	560.18	201	23	510.62
CLK-FPGA06	425K/511K/872/420	58	545	28	678.43	218	28	617.28
CLK-FPGA07	254K/309K/313/149	38	288	13	276.29	136	13	256.62
CLK-FPGA08	212K/257K/161/75	32	235	10	213.06	119	10	196.67
CLK-FPGA09	231K/358K/236/112	35	312	13	297.02	148	14	250.98
CLK-FPGA10	327K/506K/542/255	47	465	23	544.07	195	14	451.28
CLK-FPGA11	300K/468K/454/224	44	421	24	516.67	182	30	421.52
CLK-FPGA12	277K/430K/389/187	41	378	18	403.59	167	20	336.03
CLK-FPGA13	339K/405K/570/262	47	393	21	464.78	178	19	428.41
Ratio			2.251	1.037	1.128	1.000	1.000	1.000

شکل ۲-۴: خروجی های زمانی الگوریتم OpenPARF برای بنچمارک ISPD ۲۰۱۷

طراحی، زمان مراحل GP و LG و ... افزایش پیدا می‌کند و سلوول ها تا ۲ میلیون و ۱۷۷ هزار عدد افزایش داده شده است که برای این مقدار سلوول، مرحله GP ، ۴۳ ثانیه طول کشیده است.

برای الگوریتم OpenPARF هم هرچقدر دیزاین ها پیچیده تر می‌شود، هم تعداد منابع مصرفی داخلی FPGA بیشتر مصرف می‌شود و هم زمانبندی فاز چینش و مسیریابی نیز زیاد تر می‌شود.

نتیجه گیری ای که این جانب از این مطالعات دارم بدین صورت است:

با پیشرفت ابزارهای هوش مصنوعی، کم کم ابزارهای قدیمی طراحی دارند جای خود را به ابزارهای هوشمند می‌دهند. هرچند که همچنان راه زیادی وجود دارد تا جایگذینی کامل این ابزارها با هم اما بلاخره روزی فراخواهد رسید که قرایнд طراحی از مرحله طراحی شماتیک تا Layout به صورت اتوماتیک و بدون دخالت انسان انجام

می شود.

این دو مقاله به پیاده سازی ۲ مرحله مهم از طراحی، یعنی چینش قطعات و مسیر یابی آنها گام کوچ و موثری در تحقق این هدف برداشته اند.

امید است که بتوانیم با تحقیقات بیشتر و ارائه ساختارهای جدید به پیشرفت تکنولوژی در این مسیر کمک کنیم.

# Bibliography

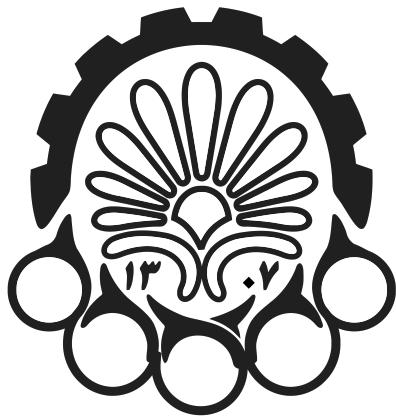
- [1] J. Mai, J. Wang, Z. Di, G. Luo, Y. Liang, and Y. Lin. Openparf: An open-source placement and routing framework for large-scale heterogeneous fpgas with deep learning toolkit. *IEEE 15th International Conference on ASIC (ASICON)*, 2023.
- [2] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan. Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement. *Proceedings of the 56th Annual Design Automation Conference*, (117):1 – 6, 2019.
- [3] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [4] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.
- [5] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.
- [6] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [7] K. E. Murray, S. Whitty, S. Liu, J. Luu, , and V. Betz. Timing-driven titan: Enabling large benchmarks and exploring the gap between academic and commercial cad. *ACM TRETS*, pages 1 – 18, 2015.

## Abstract

The main challenge of this research has been to examine and compare the efficiency of transformers against traditional architectures like U-Net in diffusion models. The obtained results indicate that transformers, with better scalability and higher efficiency, can be a suitable replacement for current architectures in diffusion models and help improve the quality of generated images. In this regard, Diffusion Transformers (DiTs), by increasing the depth and width of transformers and increasing the number of input tokens, have shown significant improvement in FID scores and achieved superior results in the ImageNet benchmark sets.

This research demonstrates that using transformers in diffusion models can provide innovative solutions for improving the quality and efficiency of image generation. These approaches, in addition to enhancing performance, can help reduce computational complexities and increase the speed of image generation processes. Given these results, transformers emerge as a strong alternative to traditional architectures in diffusion models and can be widely used in various applications.

**Keywords:** Deep Learning, Diffusion Models, Transformer, U-Net, Encoder, Decoder



Amirkabir University of Technology

(Tehran Polytechnic)

Department of Computer Engineering

Deep Learning Final Research Report Thesis

## **Diffusion Models with Transformers Neural Network**

By:

**Reza Adinepour**

Supervisor:

**Prof. Safabakhsh**

Jun 2024