به‌نام خدا

مدیریت منابع برای توابع شبکه مبتنی بر یادگیری ماشین در لایه داده برنامه‌پذیر

ارائه دهنده: فاطمه بابائی

استاد راهنما: دکتر مهدی دولتی
استاد داور داخلی: دکتر بردیا صفائی
استاد داور خارجی: دکتر ابوالفضل دیانت

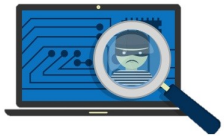
دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

شهریور ۱۴۰۴

فهرست مطالب

مقدمه

- **توابع شبکه** توابع شبکه، اجزای ضروری و ستون فقرات شبکه‌های مدرن هستند که امکان عملکرد یکپارچه، ایمن‌سازی و بهینه‌سازی ترافیک داده را فراهم کرده‌اند.
  - تشخیص ناهنجاری
  - طبقه‌بندی جریان



- **پیاده‌سازی توابع**
  - پیاده‌سازی سخت‌افزاری
  - پیاده‌سازی نرم‌افزاری
  - پیاده‌سازی با لایه داده برنامه پذیر

مفاهیم اولیه

- موضوع پژوهش، پیاده‌سازی توابع شبکه مبتنی بر یادگیری ماشین روی لایه داده برنامه‌پذیر است که شامل اجرای شبکه‌های عصبی دودویی روی آن‌ها هستند.

- این کار با توزیع وزن‌های شبکه عصبی دودویی با استفاده از برنامه‌ریزی خطی عدد صحیح و استفاده از زبان P4 انجام می‌شود.

مفاهیم اولیه

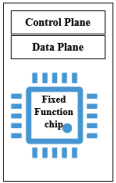
- مفاهیم اولیه در این حوزه عبارتند از:
  - زبان برنامه‌نویسی P4 (Programming Protocol-Independent Packet Processors)
  - شبکه عصبی دودویی
  - برنامه‌ریزی خطی عدد صحیح

```
control MyIngress(inout headers hdr, inout metadata meta,
                  inout standard_metadata_t standard_metadata) {
    apply {
        if (hdr.ethernet.srcAddr == 0x112233445566) {
            standard_metadata.egress_spec = 1; // Forward to port 1
        } else {
            standard_metadata.egress_spec = 2; // Forward to port 2
        }
    }
}
```

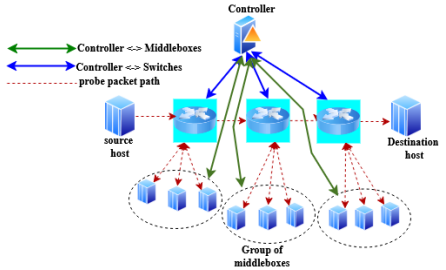
کارهای پیشین مرتبط (روش‌های استقرار)

۱ شبکه‌های سنتی

- جعبه میانی

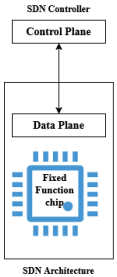


Traditional Architecture

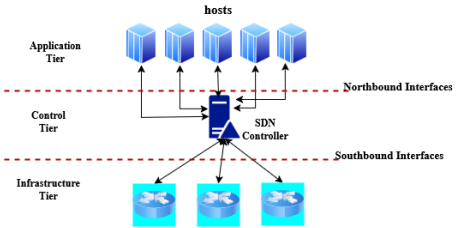


۲ شبکه‌های نرم‌افزار محور

- جداسازی صفحه کنترل و صفحه

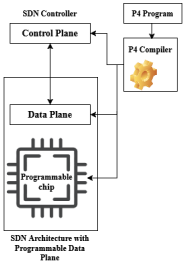


SDN Architecture



کارهای پیشین مرتبط (روش‌های استقرار)

۳ شبکه‌های برنامه‌پذیر



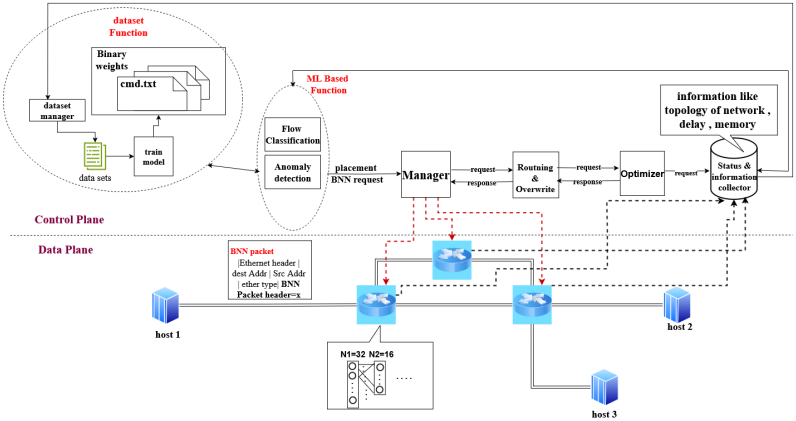
- قابلیت محاسبات درون‌شبکه‌ای
- یادگیری ماشین درون شبکه‌ای: شامل استفاده از یادگیری ماشین جهت اجرا توابع است.
  - پیاده‌سازی طبقه‌بندی که شامل تبدیل منطق برنامه‌های طبقه‌بندی یادگیری ماشین به جداول تطابق_عمل در سوئیچ است [۱] و [۲] و [۳] و [۴] و [۵] و [۶].
  - پیاده سازی شبکه عصبی که شامل استفاده از جداول تطابق_عمل و جست‌وجو است [۷] و [۸].
  - یادگیری عمیق به کمک ابزار جانبی نظیر NCS صورت گرفته و استنباط به کمک سوئیچ انجام می‌شود [۹].
  - پیاده‌سازی شبکه عصبی دودویی که شامل استفاده از قابلت‌های سوئیچ مانند XOR، AND و SHIFT است [۱۰] و [۱۱] و [۱۲] و [۱۳] و [۱۴].

مقایسه معیارهای عملکرد کارهای پیشین مرتبط

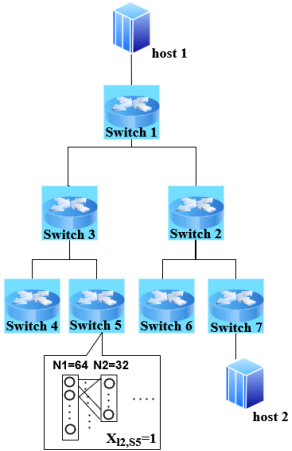
موضوع مقاله	نام مقاله	بستر پیاده‌سازی	الگوریتم یادگیری ماشین	وظیفه پردازشی	تقسیم شبکه عصبی	الگوریتم تخصیص منابع
یادگیری ماشین درون شبکه‌ای	Toward In-Network Intelligence [۷]	P4	شبکه عصبی مصنوعی	طبقه‌بندی جریان	تقسیم نورون	دارد
	IN3 [۸]	P4	شبکه عصبی مصنوعی	طبقه بندی جریان	ندارد	ندارد
	NCS [۹]	P4	شبکه عصبی کانولوشنی	طبقه‌بندی بسته	ندارد	ندارد
	Luo [۱۰]	P4	شبکه عصبی دودویی	طبقه‌بندی جریان	ندارد	ندارد
	Siracusano [۱۱]	P4	شبکه عصبی دودویی	طبقه‌بندی جریان	ندارد	ندارد
	N2net [۱۲]	P4	شبکه عصبی دودویی	طبقه‌بندی بسته‌ها	ندارد	ندارد
	Qin [۱۳]	P4	شبکه عصبی دودویی	طبقه‌بندی بسته‌ها	ندارد	ندارد
	NN-Split [۱۴]	P4	شبکه عصبی دودویی	طبقه‌بندی بسته‌ها	تقسیم لایه	ندارد
	روش پیشنهادی	**P4**	**شبکه عصبی دودویی**	**طبقه‌بندی بسته‌ها و تشخیص ناهنجاری**	**تقسیم لایه**	**دارد**

مدل سیستم

1. یادگیری توابع در صفحه کنترل
2. توزیع وزن‌های حاصل از یادگیری
   - حل مسئله بهینه‌سازی
   - مسیریابی
3. استنباط توابع به وسیله سوئیچ



تعریف مسئله



- **شبکه:** به‌صورت یک گراف $G = (V_G, E_G)$ مدل می‌شود که در آن:
  - V_G: مجموعه‌ای از سوئیچ‌های شبکه
  - E_G: مجموعه‌ی لینک‌های دوطرفه بین سوئیچ‌ها
- **متغیر تصمیم:** متغیر تصمیم $X_{l_i,s}$ به‌صورت زیر تعریف می‌شود:

$$X_{l_i,s} \in \{0, 1\}$$

که در آن:

- اگر $X_{l_i,s} = 1$: لایه‌ی i روی سوئیچ s مستقر شده است.
- اگر $X_{l_i,s} = 0$: لایه‌ی i روی سوئیچ s مستقر نشده است.

تعریف مسئله

هدف کلی: یافتن استقراری که:

- بهره‌وری منابع شبکه را حداکثر کند
- توزیع لایه‌ها را مطابق با محبوبیت آن‌ها نگه دارد
- تأخیر ارتباطی بین لایه‌های متوالی شبکه‌های عصبی دودویی را به حداقل برساند

$$\max \gamma = \underbrace{\sum_{l \in layer} n_l}_{\text{بهره‌وری منابع}} - \tau_1 \underbrace{\sum_{l \in layer} \left| \frac{n_l}{N} - \hat{p}_l \right|}_{\text{انحراف از توزیع محبوبیت}} - \tau_2 \underbrace{\sum_{BNN_t \in BNN} \sum_{l_i, l_j \in BNN_t} d\Big(\sum_{s_k \in V_G} X_{l_i, s_k} s_k, \sum_{s_m \in V_G} X_{l_j, s_m} s_m \Big)}_{\text{تأخیر بین‌لایه‌ای}}$$

- τ_1, τ_2: ضرایب تنظیم‌پذیر برای تعادل اهداف چندگانه

تعریف مسئله

هدف ۱: بهره‌وری از منابع همراه با تخصیص موثر لایه‌ها با توجه به محبوبیت آن‌ها.

$$\lambda = \sum_{l \in layer} n_l - \tau_1 \sum_{l \in layer} \left| \frac{n_l}{N} - \hat{p}_l \right|$$

- n_l: تعداد دفعات ظهور لایه l در کل شبکه‌های عصبی دودویی
- N: مجموع ظهور کل لایه‌ها
- $\hat{p}_l$: محبوبیت نرمال شده لایه l
- τ_1: ضریب تنظیم‌پذیر برای کنترل جریمه انحراف از توزیع محبوبیت

تعریف مسئله

هدف ۲: کاهش تأخیر هنگام انتقال داده‌ها بین لایه‌های متوالی شبکه عصبی دودویی.

$$\beta = \sum_{BNN_t \in BNN} \sum_{(l_i,l_j) \in BNN_t} d\left(\sum_{s_k \in V_G} X_{l_i,s_k} s_k, \sum_{s_m \in V_G} X_{l_j,s_m} s_m \right)$$

- $d\left(X_{l_i,s_k} \cdot s_k,\ X_{l_j,s_m} \cdot s_m\right)$: میزان تأخیر بین دو لایه متوالی l_i و l_j روی سوئیچ‌های s_k و s_m
- معیار محاسبه تاخیر در این پژوهش: تعداد پرش‌ها (Hop-Count)

تعریف مسئله

- حافظه مصرفی توسط لایه‌ها نباید از ظرفیت SRAMسوئیچ بیشتر باشد.

$$\sum_{BNN_t \in BNN} \sum_{l_i \in BNN_t} X_{l_i,s} \cdot SRAM(l_i) \leq N_{SRAM}$$

- مجموع مصرف PHV توسط لایه‌ها نباید از ظرفیت PHV سوئیچ بیشتر شود.

$$\sum_{BNN_t \in BNN} \sum_{l_i \in BNN_t} X_{l_i,s} \cdot PHV(l_i) \leq N_{PHV}$$

- مصرف واحدهای محاسباتی نباید از ظرفیت موجود تجاوز کند.

$$\sum_{BNN_t \in BNN} \sum_{l_i \in BNN_t} X_{l_i,s} \cdot SLA(l_i) \leq N_{SLA}$$

- هر لایه باید حداقل روی یک سوئیچ مستقر شود.

$$\sum_{s \in V_G} X_{l_i,s} \geq 1$$

این محدودیت‌ها تضمین می‌کنند که تخصیص لایه‌ها به سوئیچ‌ها ضمن رعایت محدودیت‌های سخت‌افزاری، عملیاتی باقی بماند.

روش پیشنهادی

- مسئله‌ی تخصیص لایه‌های شبکه‌های عصبی بر روی سوئیچ‌ها، مشابه مسئله‌ی تخصیص عمومی (GAP) بوده و از نظر محاسباتی جزو مسائل NP-Hard محسوب می‌شود.

- برای حل این مسئله از رویکرد **ملایم‌سازی (Relaxation)** همراه با **روش‌های تقریبی (Approximation)** استفاده شده است.

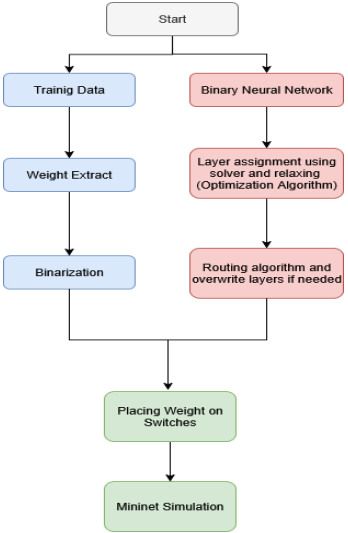
- در مرحله‌ی ملایم‌سازی، محدودیت‌های عدد صحیح حذف شده و مسئله به یک برنامه ریزی خطی (Linear Programming) تبدیل می‌شود و توسط حل‌کننده SCIP حل می‌شود. سپس، برای تبدیل جواب کسری به جواب صحیح، از الگوریتم‌های تقریبی استفاده می‌شود تا یک راه‌حل عملی و زیر بهینه به‌دست آید.

الگوریتم تقریبی

الگوریتم ۱: الگوریتم پساپردازش برای تخصیص لایه‌ها

مرحله ۱: تخصیص اولیه ؛
برای هر لایه ؛
مرتب‌سازی سوئیچ‌ها بر اساس مقدار حل‌کننده به صورت نزولی ؛
پیدا کردن سوئیچ با کمترین امتیاز ($score(l,s) \leftarrow d(l,s) - \alpha \cdot p\_hat[l]$) ؛
اختصاص لایه به آن سوئیچ و به‌روزرسانی منابع؛
مرحله ۲: توسعه تخصیص ؛
مرتب‌سازی لایه‌ها بر اساس محبوبیت به صورت نزولی ؛
مرتب‌سازی سوئیچ‌ها بر اساس مقدار حل‌کننده به صورت نزولی ؛
برای هر لایه ؛
برای هر سوئیچ ؛
اگر لایه روی سوئیچ نیست و منابع کافی موجود است؛
محاسبه تأثیر افزایش تأخیر ؛
اگر تأثیر مجاز است: افزودن لایه به سوئیچ و به‌روزرسانی منابع ؛
مرحله ۳: تضمین تخصیص ؛
برای هر لایه ؛
اگر لایه به سوئیچی تخصیص داده نشده: تلاش برای اختصاص به بهترین سوئیچ بر اساس مقادیر حل‌کننده به صورت نزولی ؛
اگر هیچ سوئیچی منابع کافی ندارد: حذف لایه‌های کم‌محبوب و بیش از یک بار تخصیص داده شده و در ادامه تخصیص لایه فعلی ؛

پیاده‌سازی عملی



فاز یادگیری

- آماده‌سازی داده‌های آموزشی
- آموزش مدل بر روی شبکه عصبی دودویی
- استخراج وزن‌ها
- دودویی‌سازی (Binarization)

فاز تخصیص و مسیریابی

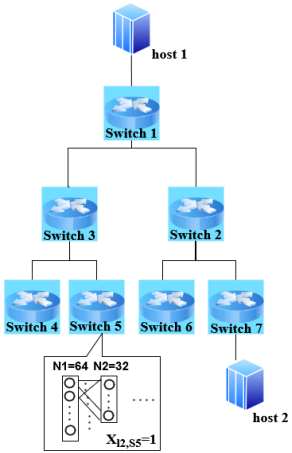
- دریافت شبکه عصبی دودویی
- تخصیص لایه‌ها (Relaxation & Approximation)
- پس‌پردازش
  - مسیریابی
  - افزودن یا بازنویسی لایه در صورت نیاز

فاز استقرار

- یکپارچه‌سازی نهایی
- قرار دادن وزن‌ها روی سوئیچ‌ها

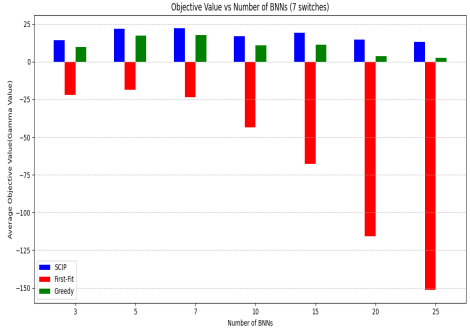
تنظیمات ارزیابی

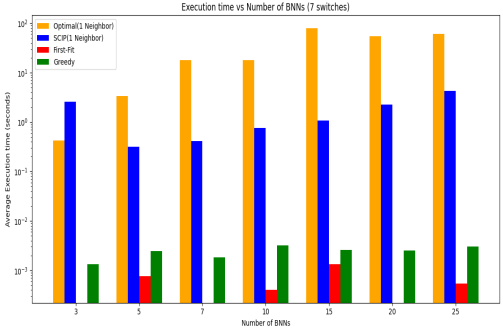
پارامتر	تنظیمات
توپولوژی شبکه	توپولوژی Fat-tree با ضریب انشعاب ۲
تعداد سوئیچ‌ها	۷
تعداد میزبان‌ها	۲
تعداد شبکه‌های عصبی دودویی	۳ تا ۲۵
تعداد لایه‌های شبکه عصبی دودویی	۳ تا ۵
تعداد نورون در هر لایه	۸ تا ۶۴
محدوده منابع سوئیچ‌ها	بر اساس Tofino 2 (مانند Edgecore Wedge100BF-32X)
زبان پیاده‌سازی الگوریتم	Python
مجموعه داده‌ها	دسته‌بندی اینترنت اشیا (IoT Analytics) و تشخیص حمله (UNSW-NB15)
ضریب تنظیم پذیر γ_1 در برنامه ریزی خطی	۰.۱
ضریب تنظیم پذیر γ_2 در برنامه ریزی خطی	۱۰
معیار محاسبه تأخیر	تعداد پرش ها (Hop-Count)
حد مجاز افزایش تاخیر در الگوریتم تقریبی	۱۰ واحد
سیاست مسیریابی	مشاهده تمام لایه‌های مورد نیاز در حداکثر ۱.۳ برابر کوتاه‌ترین مسیر
محیط شبیه‌سازی شبکه	Mininet روی لینوکس
نسخه پایتون شبیه‌سازی	۳.۸.۱۰



نتایج الگوریتم

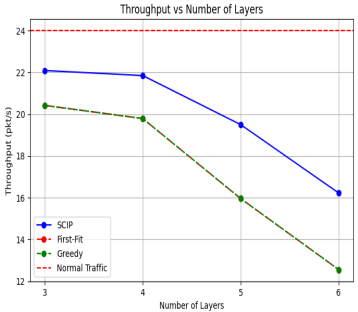
- با افزایش تعداد شبکه‌های عصبی مقدار تابع هدف همواره روش پیشنهادی بهتر از روش‌های پایه عمل کرده است.
- در مقایسه با روش حریصانه، این بهبود برای کمتر از ۱۰ شبکه عصبی حدود ۲ تا ۴ برابر و برای بیش از ۱۰ شبکه عصبی حدود ۱۲ برابر بوده است. ولی در روش اولین برازش در کمتر از ۱۰ شبکه عصبی این مقدار ۱.۱ برابر منفی شده و بیش از ۱۰ شبکه عصبی حدود ۵ برابر منفی شده است.
- در مقایسه با روش بهینه، روش پیشنهادی برای کمتر از ۱۰ شبکه عصبی حدود ٪۸۹ کاهش زمان اجرا داشته است و برای بیش از ۱۰ شبکه عصبی این کاهش به حدود ٪۹۶ رسیده است.

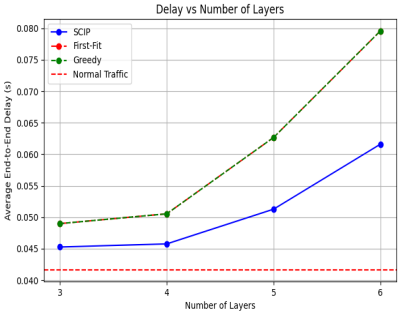




نتایج شبیه‌سازی

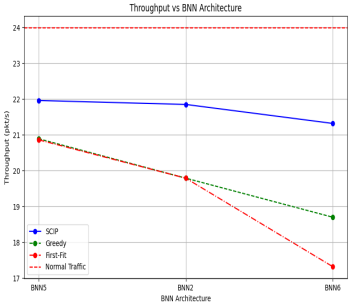
- با افزایش تعداد لایه نرخ گذردهی کاهش و تأخیر انتها به انتها افزایش می‌یابد.
- نرخ گذردهی و تأخیر انتها به انتها در تغییر لایه در روش پیشنهادی بهتر از روش‌های پایه است.

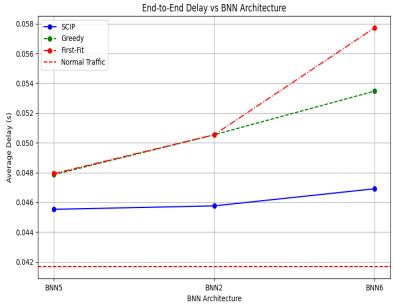




نتایج شبیه‌سازی

- با افزایش تعداد نورون در هر لایه نرخ گذردهی کاهش و تأخیر انتها به انتها افزایش می‌یابد.
- نرخ گذردهی و تأخیر انتها به انتها در تغییر تعداد نورون در هر لایه در روش پیشنهادی بهتر از روش‌های پایه است.





جمع‌بندی

- این پژوهش به بهبود عملکرد و کارایی شبکه‌ها با ادغام یادگیری ماشین در زیرساخت شبکه و با توجه به منابع محدود می‌پردازد.
- هدف این رویکرد کاهش تأخیر، بهبود توان عملیاتی و فعال‌سازی مدیریت هوشمند و مستقل شبکه است.
- رویکرد پیشنهادی شامل دو مرحله است:

- **فاز یادگیری:** شامل سه مرحله است. در مرحله اول، توابع شبکه با استفاده از کنترل‌کننده مرکزی و مدل‌هایی مانند شبکه‌های عصبی دودویی آموزش می‌بینند و وزن‌های هر لایه استخراج شده و به وزن‌های دودویی تبدیل می‌شوند. در مرحله دوم، لایه‌های مدل یادگیری ماشین بر اساس منابع موجود در سراسر شبکه توزیع می‌شود. این مسئله به عنوان مسئله برنامه‌ریزی خطی عدد صحیح فرموله می‌شود و با استفاده از اطلاعات شبکه مانند حافظه و واحدهای منطق حل می‌شود. در مرحله سوم بر اساس تخصیص انجام شده مسیریابی می‌شود و در صورت نیاز برخی لایه‌هابازنویسی می‌شود.

- **فاز استنباط:** با استفاده از برنامه‌های P4 در دستگاه‌های شبکه (سوئیچ‌ها) اجرا می‌شود. این برنامه‌ها توابع شبکه آموخته‌شده را مستقیماً در صفحه داده پیاده‌سازی کرده و در مورد ادامه روند بسته تصمیم‌گیری می‌کنند.

مراجع ۱

[1] C. Zheng, Z. Xiong, T. Bui, S. Kaupmees, R. Bensoussane, A. Bernabeu, S. Vergaftik, Y. Ben-Itzhak, and N. Zilberman, "Iisy: Hybrid in-network classification using programmable switches," IEEE/ACM ToN, 32(3):2555–2570, 2024.

[2] Z. Xiong and N. Zilberman, "Do switches dream of machine learning? toward in-network classification," in HotNets, 2019.

[3] X. Zhang, L. Cui, F. P. Tso, and W. Jia, "pheavy: Predicting heavy flows in the programmable data plane," IEEE TNSM, 18(4):4353–4366, 2021.

[4] R. Kamath and K. M. Sivalingam, "Machine learning based flow classification in dcns using p4 switches," in ICCCN, pp.1–10, 2021.

[5] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted ddos attack detection with p4 language," in ICC, pp.1–6, IEEE, 2020.

[6] Q. Li, J. Lin, G. Xie, Z. Guan, Z. Luan, and Z. Qi, "Distributed multi-task in-network classification on programmable switches by ensemble models," IEEE ToN, pp.1–16, 2025.

[7] M. Saquetti, R. Canofre, A. Lorenzon, F. D. Rossi, J. Azambuja, W. Cordeiro, and M. C. Luizelli, "Toward in-network intelligence: Running distributed artificial neural networks in the data plane," IEEE Communications Letters, 25(11):3551–3555, 2021.

[8] X. Zhang, L. Cui, F. P. Tso, W. Li, and W. Jia, "In3: A framework for in-network computation of neural networks in the programmable data plane," IEEE Communications Magazine, 62(4):96–102, 2024.

[9] Y.-S. Lu and K.-J. Lin, "Enabling inference inside software switches," in APNOMS, pp.1–4, IEEE, 2019.

[10] J. Luo, W. Liu, M. Tan, and H. Chen, "Binary neural network with p4 on programmable data plane," in MSN, 2022.

[11] G. Siracusano, S. Galea, D. Sanvito, M. Malekzadeh, G. Antichi, P. Costa, H. Haddadi, and R. Bifulco, "Re-architecting traffic analysis with neural network interface cards," in NSDI, 2022.

[12] G. Siracusano and R. Bifulco, "In-network neural networks," arXiv:1801.05731, 2018.

[13] Q. Qin, K. Poularakis, K. Leung, and L. Tassiulas, "Line-speed and scalable intrusion detection at the network edge via federated learning," in IFIP Networking, pp.352–360, IEEE, 2020.

مراجع ۲

[14] D. C. Li, M. R. Maulana, and L.-D. Chou, "Nnsplit-sØren: Supporting the model implementation of large neural networks in a programmable data plane," Computer Networks, 222:109537, 2023.

[15] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," IEEE Communications Surveys & Tutorials, 18(1):236–262, 2016.

[16] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," Proceedings of the IEEE, 103(1):14–76, Jan 2015.

[17] E. I. S. G. (ISG), "Network functions virtualisation (nfv): Architectural framework," ETSI GS NFV 002 V1.1.1, pp.1–21, 2013.

[18] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," ACM SIGCOMM CCR, 44(3):87–95, 2014.

[19] C. Zheng, X. Hong, D. Ding, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "In-network machine learning using programmable network devices: A survey," IEEE Communications Surveys & Tutorials, 28(2):1171–1200, 2023.

[20] C. Zheng, M. Zang, X. Hong, R. Bensoussane, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "Automating in-network machine learning," arXiv preprint arXiv:2205.08824, 2022.

[21] J. Xie, D. Guo, Z. Hu, and T. Qu, "Control plane of software defined networks: A survey," Computer Communications, pp.1–15, 2015.

[22] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," IEEE Access, 9:87094–87155, 2021.

[23] N. Gebara, A. Lerner, M. Yang, M. Yu, P. Costa, and M. Ghobadi, "Challenging the stateless quo of programmable switches," in HotNets, 2019.

[24] Y. Gao and Z. Wang, "A review of p4 programmable data planes for network security," Mobile Information Systems, pp.1–24, 2021.

مراجع ۳

[25] L. A. Wolsey, “Integer Programming,” John Wiley & Sons, 1998.

[26] T. Benson, A. Akella, and D. A. Maltz, “The road to sdn: An intellectual history of programmable networks,” ACM Queue, 11(12):1–21, 2013.

[27] S. Kianpisheh and T. Taleb, “A survey on in-network computing: Programmable data plane and technology specific applications,” IEEE Communications Surveys & Tutorials, 25(1):701–728, 2023.

[28] Y. Li and M. Chen, “Software-defined network function virtualization: A survey,” IEEE Access, 3:2542–2553, 2015.

[29] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” PhD Thesis, UC Irvine, 2000.

[30] S. Saraswat, V. Agarwal, H. P. Gupta, R. Mishra, A. Gupta, and T. Dutta, “Challenges and solutions in software defined networking: A survey,” Journal of Network and Computer Applications, 141:23–58, 2019.

[31] W. Xu, Z. Zhang, Y. Feng, H. Song, Z. Chen, W. Wu, G. Liu, Y. Zhang, S. Liu, Z. Tian, and B. Liu, “Clickinc: In-network computing as a service in heterogeneous programmable data-center networks,” in SIGCOMM, pp.798–815, 2023.

[32] C. Zheng, H. Tang, M. Zang, X. Hong, A. Feng, L. Tassiulas, and N. Zilberman, “Dinc: Toward distributed in-network computing,” ACM Networking, 1(14):1–25, 2023.

[33] N. Sultana, J. Sonchack, H. Giesen, I. Pedisich, Z. Han, N. Shyamkumar, S. Burad, A. DeHon, and B. T. Loo, “Flightplan: Dataplane disaggregation and placement for p4 programs,” in NSDI, pp.571–592, 2021.

[34] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, “Deep packet: A novel approach for encrypted traffic classification using deep learning,” Soft Computing, 24(3):1999–2012, 2020.

[35] X. Chen, Q. Xiao, H. Liu, Q. Huang, D. Zhang, X. Liu, L. Hu, H. Zhou, C. Wu, and K. Ren, “Eagle: Toward scalable and near-optimal network-wide sketch deployment in network measurement,” in SIGCOMM, pp.291–310, 2024.

مراجع ۴

[36] A. Drexl and K. Jørnsten, “Pricing the generalized assignment problem,” Manuskript 627, Universität Kiel, 2007.

[37] S. Arora, “Provable approximation via linear programming,” Lecture Notes COS521, Princeton, 2013.

[38] T. Achterberg, “Scip: Solving constraint integer programs,” Math. Prog. Computation, 1(1):1–41, 2009.

[39] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: Rapid prototyping for software-defined networks,” in HotNets, pp.1–6, 2010.

[40] P4.org, “Behavioral model (bmv2): The reference p4 software switch,” 2019. Accessed: 2025-09-04.

[41] Edgecore Networks, “Wedge 100-32x product specifications,” https://www.edge-core.com/, 2023. Accessed: 29 Aug 2025.

[42] P4.org, “Simple switch cli documentation,” 2019. Accessed: 2025-09-04.

[43] A. Hagberg, D. Schult, and P. Swart, “Exploring network structure, dynamics, and function using networkx,” in SciPy, pp.11–15, 2008.

با تشکر از توجه شما