NoProp:
Трениране на невронни мрежи без Back-propagation или Forward-propagation

Даниел Халачев, 4MI3400603, ИИОЗ

Цветелина Чакърова, 8MI3400591, ИИОЗ

Николета Бейска, 2MI3400639, ИИОЗ

Ограничения на съвременните подходи за машинно самообучение

- Последователна зависимост - forward-propagation трябва да завърши преди back-propagation може да се изпълни
- Високи изисквания за памет за изчисляване на градиенти
- Трудност при паралелизация
- Catastrophic forgetting - последователният характер води до загуба на контекст

Въпреки десетилетия изследвания за решаване на тези проблеми, до момента нито един метод не надминава градиентната оптимизация по точност и стабилност.

Подходът NoProp

- Нов подход за трениране на невронни мрежи, вдъхновен от дифузионни модели
- Всеки слой се обучава независимо без forward-propagation и back-propagation през целия модел.
- Намалява изискванията за памет
- Позволява паралелна обработка
- Ускорява обучението, като същевременно запазва или подобрява точността на класификация в сравнение с традиционните методи
- Три варианта: NoProp-DT (дискретно време), NoProp-CT (непрекъснато време) и NoProp-FM (flow matching)

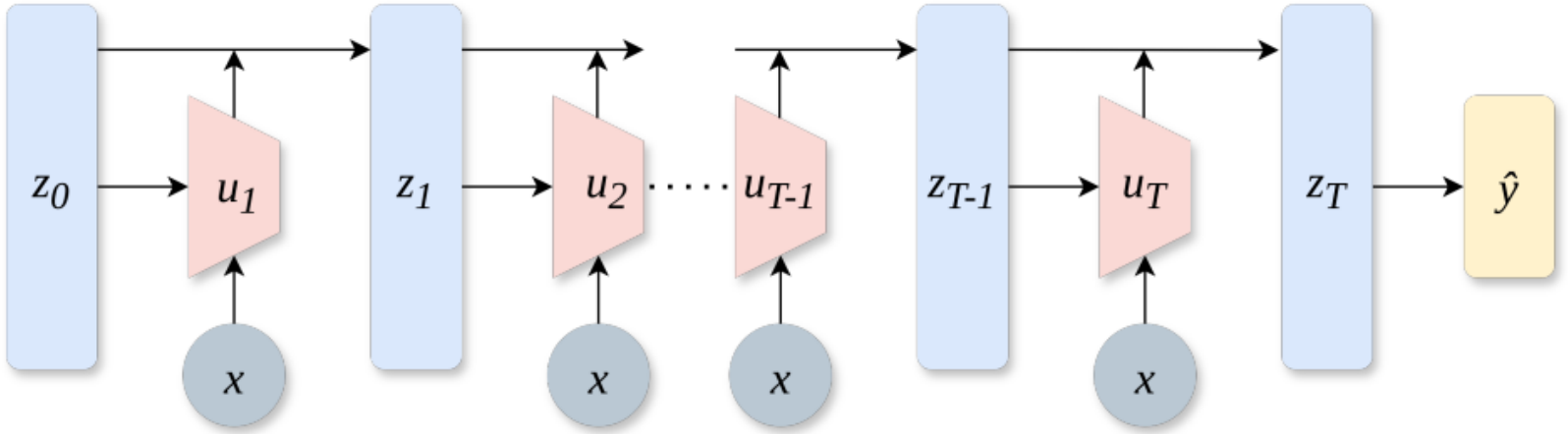
Цели на проекта

- Имплементация на NoProp-DT и NoProp-CT
- Тестване върху MNIST, CIFAR-10, CIFAR-100
- Тестване върху набор от данни с по-висока резолюция - BLOODMNIST (128x128)
- Сравнение с резултатите от оригиналната статия

Архитектура на NoProp: Фаза на извод

- Входните данни се обработват в серия от стъпки - трансформации на данните, чрез прекарването им през **дифузионени блокове.**
- Вход на мрежата: начално изображение **x** и **гаусов шум z_0**
- На **всяка стъпка** от този процес:
  - Създава се **латентна променлива z_t**, получена чрез **дифузионен блок u_t**
  - Този блок приема:
    - предишната латентна променлива z_{t-1}
    - входното изображение **x**
  - След **t на брой трансформации** получаваме последователност от междинни представяния: $z_1, z_2, \ldots, z_t$
- Последната латентна променлива z_t се изпраща към **линеен слой**, последван от **softmax функция**, за да се направи **класификация**

Архитектура на NoProp: Фаза на извод



Архитектура на фазата на извод. z_1, z_2, . . . , z_T са последователните етапи на трансформация на първоначалния шум z_0. u_1, u_2, . . . u_T са дифузионните блокове, които се обучават да премахват различни нива на шум. Линейният слой и softmax активиращата функция не са изобразени експлицитно.

Архитектура на NoProp: Фаза на обучение

- Всеки блок $\mathbf{u_t}$ се разглежда като единична невронна мрежа, която получава на входа си изображението $\mathbf{x}$, и се тренира самостоятелно от останалите блокове
  - Позволява по-лесна паралелизация
  - Намалява изискванията за памет

- Линейният слой, използван за класификация, се тренира едновременно с всички блокове
  - Осигурява се съгласуваност на научените характеристики
  - Предотвратява се срив в ембедингите на класовете

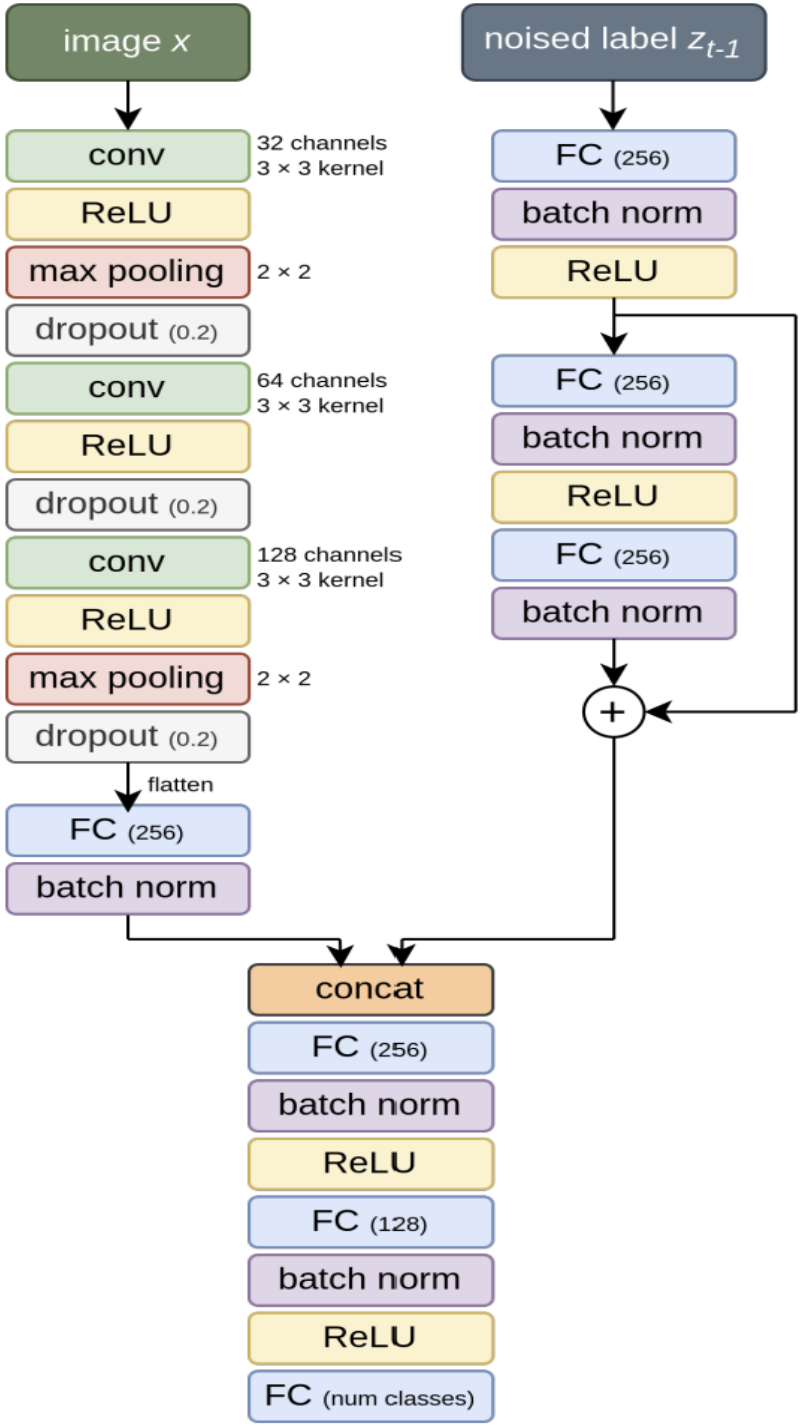
Архитектура на NoProp: Структура на дифузионния блок

- Общи структурни сегменти за NoProp-DT и NoProp-CT:
  - Кодиране на изображението
    - Конволюционни слоеве
    - Напълно свързан слой
  - Кодиране на предходния етикет
  - Конкатениращ сегмент

- В допълнение, блокът на NoProp-CT включва и сегмент за кодиране на времевите последователности.

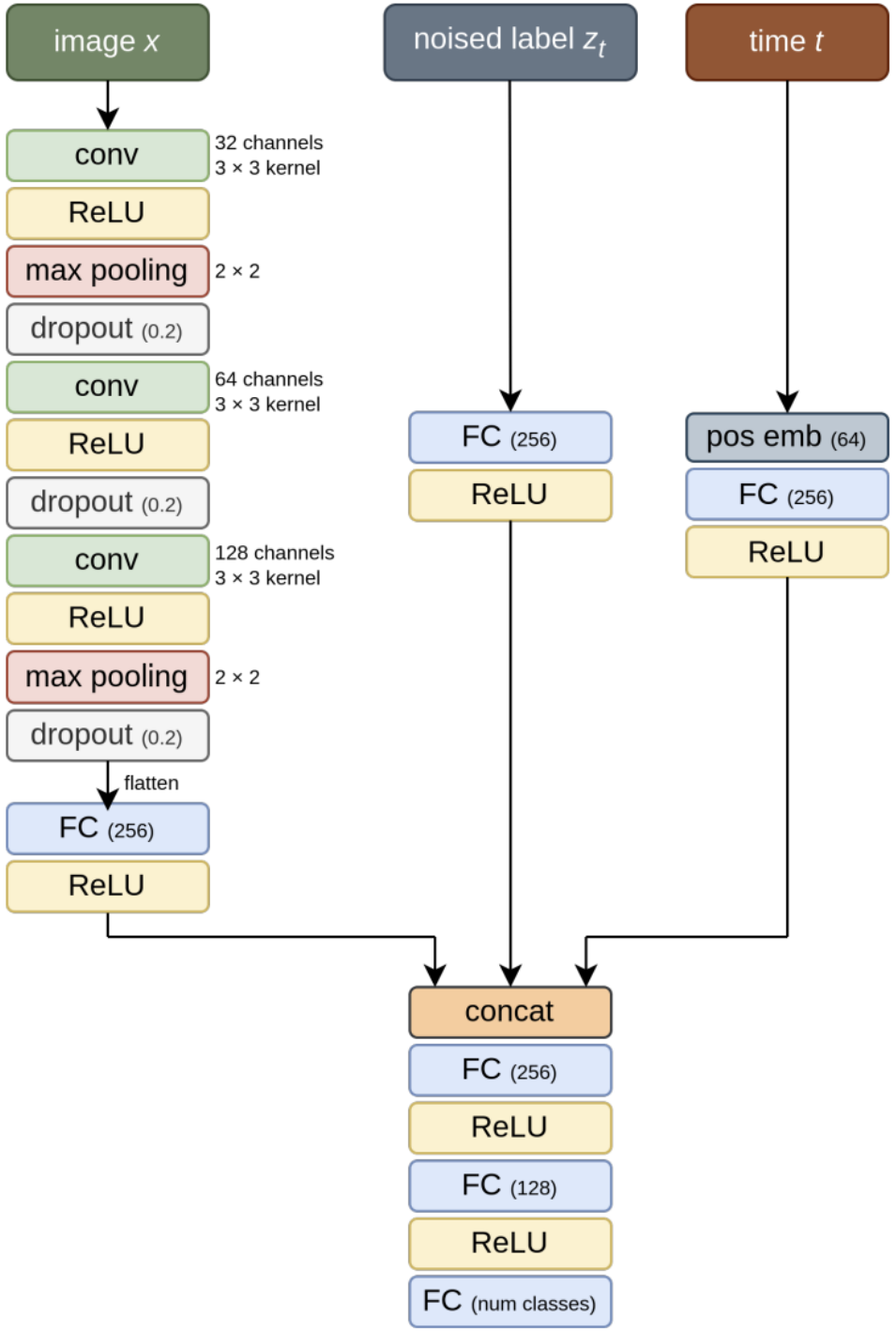
Архитектура на NoProp: NoProp-DT

- Изображението се обработва в краен брой стъпки T , които са параметър на модела.



Архитектура на NoProp: NoProp-CT

- Фазата на трениране е идентична с тази на NoProp-DT със следната разлика:
  - На входа, освен изображението x, се подава и времеви параметър t - кодира времевата зависимост чрез positional embeddings.
  - Всеки блок u_t приема и времевия параметър t.
  - Всяка поетапна трансформация се адаптира според етапа на дифузия - моделът "знае" колко е напреднал процесът.



Псевдокод на тренирането: NoProp-DT

Algorithm 1 NoProp-DT (Training)

Require: T diffusion steps, dataset $\{(x_i, y_i)\}_{i=1}^{N}$, batch size B, hyperparameter η, embedding matrix W_{Embed}, parameters $\{\theta_t\}_{t=1}^{T}, \theta_{\text{out}}$, noise schedule $\{\alpha_t\}_{t=0}^{T}$

for $t = 1$ to T **do**

  for each mini-batch $\mathcal{B} \subset \{(x_i, y_i)\}_{i=1}^{N}$ of size B **do**

    for each $(x_i, y_i) \in \mathcal{B}$ **do**

      Obtain label embedding $u_{y_i} = \{W_{\text{Embed}}\}_{y_i}$.

      Sample $z_{t,i} \sim \mathcal{N}_d(z_{t,i} | \sqrt{\bar{\alpha}_t} u_{y,i}, 1 - \bar{\alpha}_t)$.

    end for

    Compute the loss function:

$$\mathcal{L}_t = \frac{1}{B} \sum_{i \in \mathcal{B}} [-\log \hat{p}_{\theta_{\text{out}}}(y_i | z_{T,i})] + \frac{1}{B} \sum_{i \in \mathcal{B}} D_{\text{KL}}(q(z_0 | y_i) \| p(z_0)) + \frac{T}{2B} \eta \sum_{i \in \mathcal{B}} (\text{SNR}(t) - \text{SNR}(t-1)) \left\| \hat{u}_{\theta_t}(z_{t-1,i}, x_i) - u_{y_i} \right\|^2 .$$

    Update θ_t, θ_{out}, and W_{Embed} using gradient-based optimization.

  end for

end for

Псевдокод на тренирането: NoProp-CT

Algorithm 2 NoProp-CT (Training)

Require: dataset $\{(x_i, y_i)\}_{i=1}^N$, batch size B, hyperparameter η, embedding matrix W_{Embed}, parameters $\theta, \theta_{\text{out}}$, noise schedule $\bar{\alpha}_t = \sigma(-\gamma_\psi(t))$

for each mini-batch $\mathcal{B} \subset \{(x_i, y_i)\}_{i=1}^N$ with size B **do**

for each $(x_i, y_i) \in \mathcal{B}$ **do**

Obtain label embedding $u_{y_i} = \{W_{\text{Embed}}\}_{y_i}$.

Sample $t_i \sim \mathcal{U}(0, 1)$.

Sample $z_{t_i,i} \sim \mathcal{N}_d(z_{t_i,i} | \sqrt{\bar{\alpha}_{t_i}} u_{y,i}, 1 - \bar{\alpha}_{t_i})$.

end for

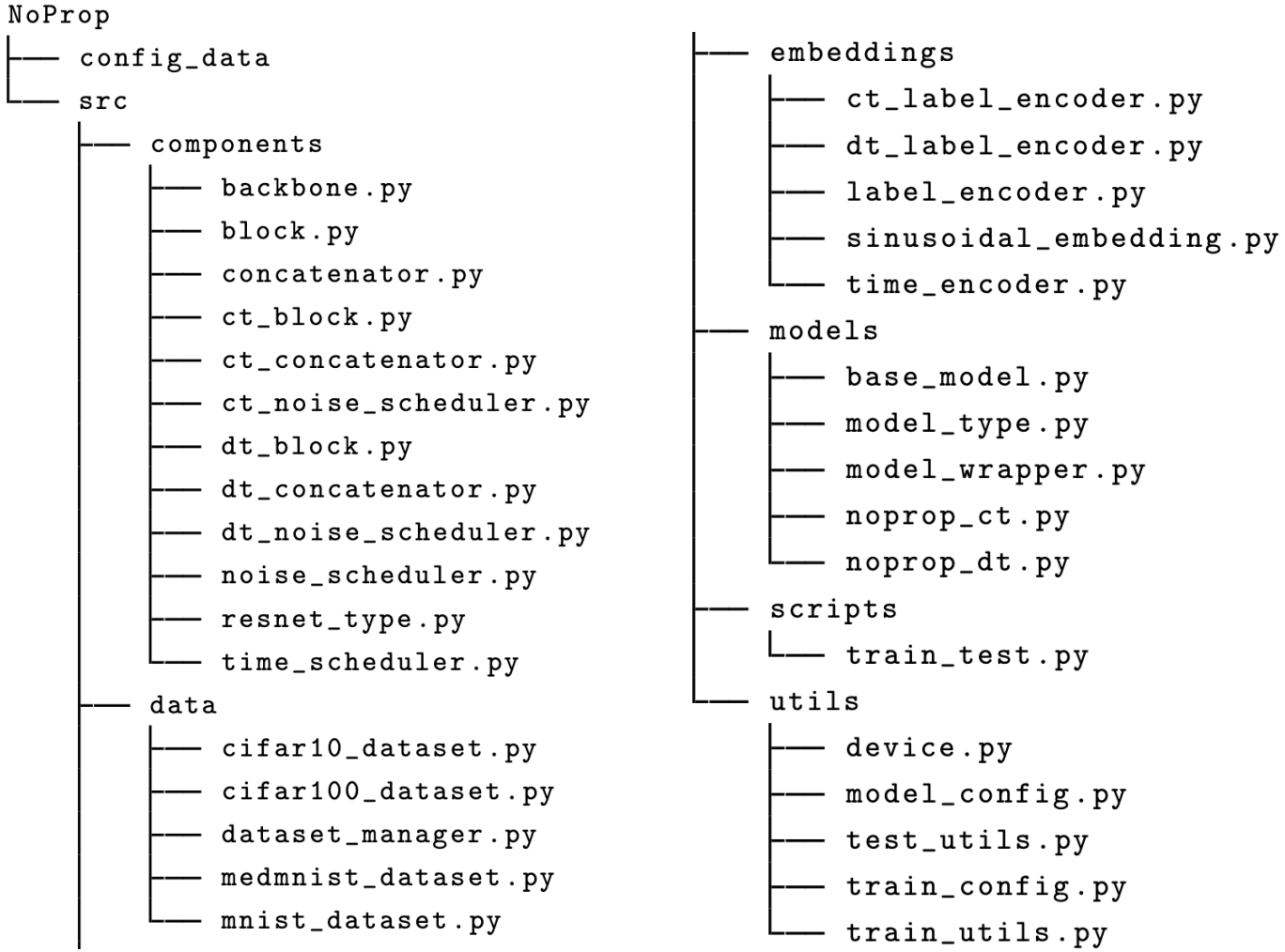
Compute the loss function:

$$\begin{aligned}\mathcal{L} = &\frac{1}{B} \sum_{i \in \mathcal{B}} [-\log \hat{p}_{\theta_{\text{out}}}(y_i | z_{1,i})] \\ &+ \frac{1}{B} \sum_{i \in \mathcal{B}} D_{\text{KL}}(q(z_0 | y_i) \| p(z_0)) \\ &+ \frac{1}{2B} \eta \sum_{i \in \mathcal{B}} \text{SNR}'(t_i) \, \| \hat{u}_\theta(z_{t_i,i}, x_i, t_i) - u_{y_i} \|^2 .\end{aligned}$$

Update θ, θ_{out}, ψ, and W_{Embed} using gradient-based optimization.

end for

Структура на реализацията



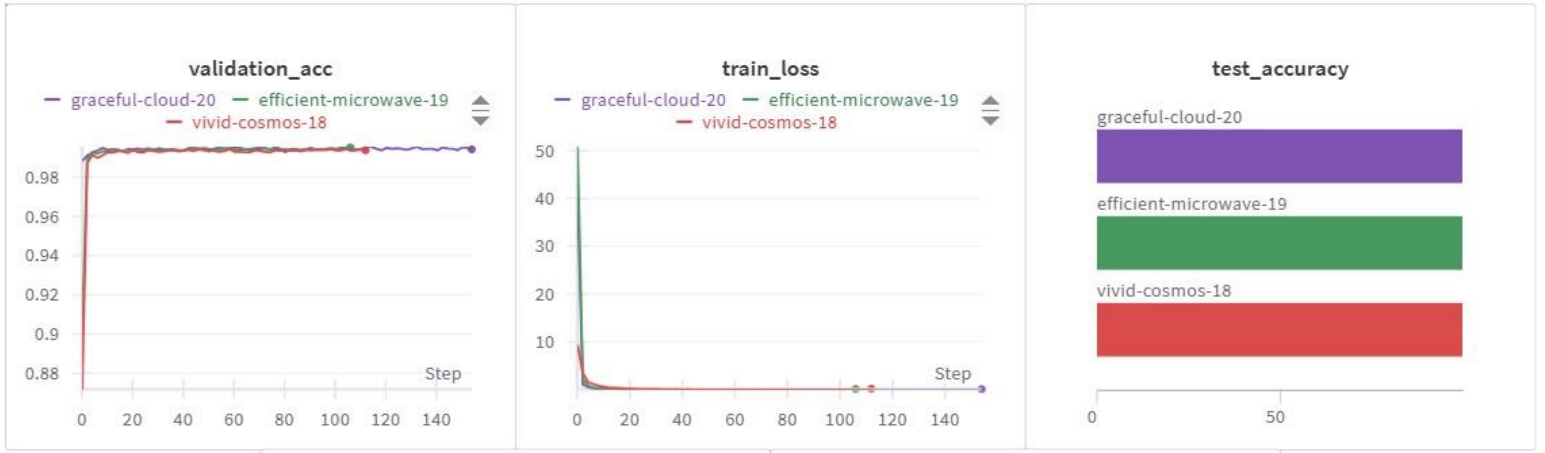
Използвани стойности на хиперпараметрите

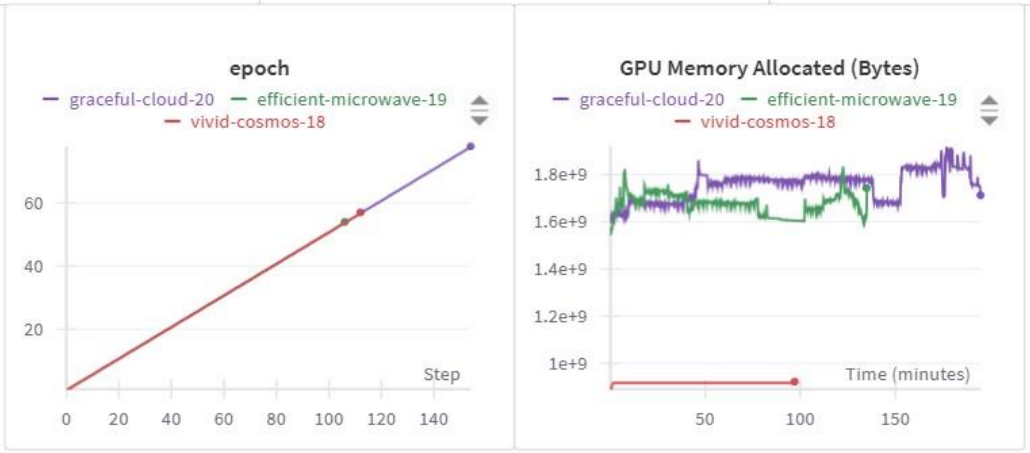
Набор	**Вариант**	**Бач**	**Епохи**	**Опт.**	**lr**	**wd**	**Стъпки**	η
MNIST	NoProp-DT	128	100	AdamW	0.001	0.001	10	0.1
CIFAR-10	NoProp-DT	128	150	AdamW	0.001	0.001	10	0.1
CIFAR-100	NoProp-DT	128	150	AdamW	0.001	0.001	10	0.1
BLOODMNIST	NoProp-DT	128	100	AdamW	0.001	0.001	1000	0.1
MNIST	NoProp-CT	128	100	Adam	0.001	0.001	1000	1
CIFAR-10	NoProp-CT	128	500	Adam	0.001	0.001	1000	1
CIFAR-100	NoProp-CT	128	1000	Adam	0.001	0.001	1000	1
BLOODMNIST	NoProp-CT	128	1000	Adam	0.001	0.001	1000	1

Таблица 2: Стойности на хиперпараметри за различни множества от данни и алгоритми. `lr` обозначава `learning_rate`, `wd` - `weight_decay`, а стъпките се отнасят до фазата на извод.

Експерименти - NoProp-DT и MNIST

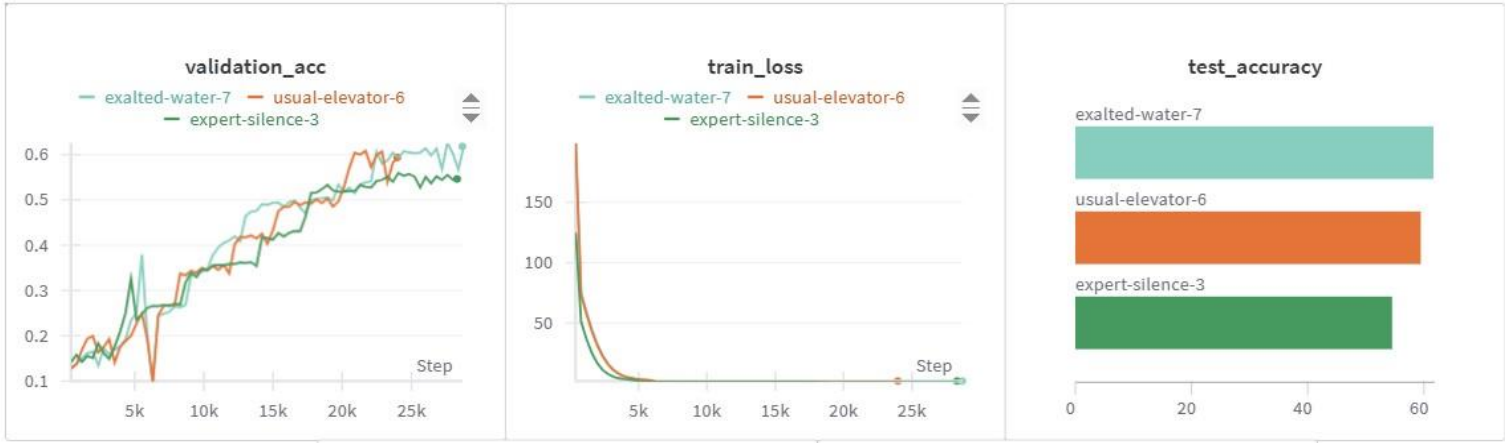
Име на изпълнение	Загуба	Тестова точност	Брой епохи	GPU памет (GB)
graceful-cloud-20	0.0875	99.42	78	1.92
efficient-microwave-19	0.1151	99.53	54	1.84
vivid-cosmos-18	0.1743	99.37	57	0.86





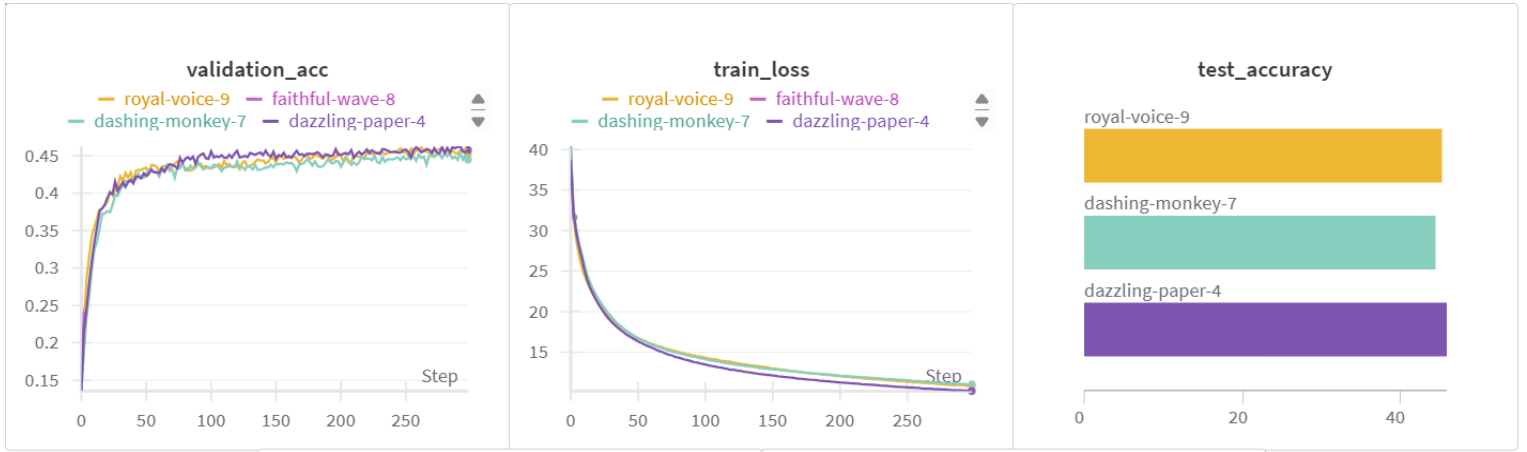
Експерименти - NoProp-DT и CIFAR-10

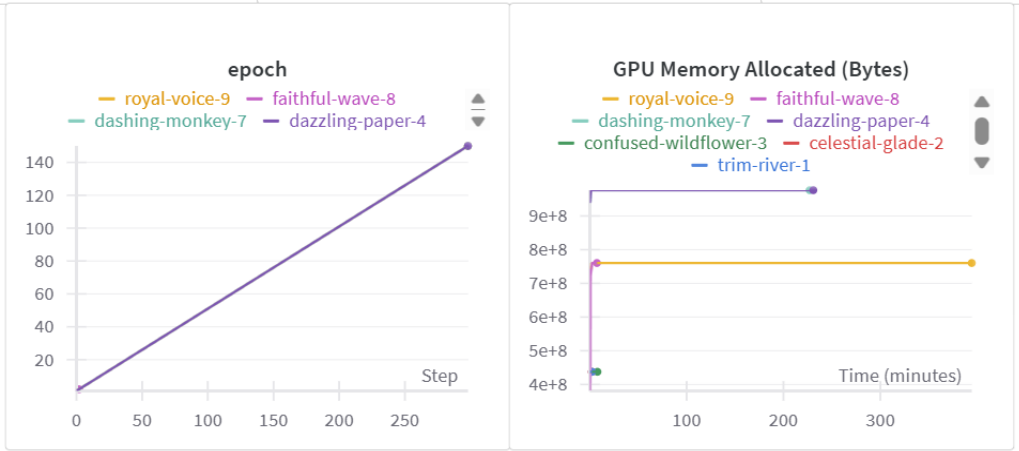
Име на изпълнение	Загуба	Тестова точност	Брой епохи	GPU памет (GB)
helpful-gorge-6	1.3370	80.54	150	0.69
curious-waterfall-5	1.4325	79.62	150	1.84
fancy-terrain-1	1.2189	80.75	150	0.91



Експерименти - NoProp-DT и CIFAR-100

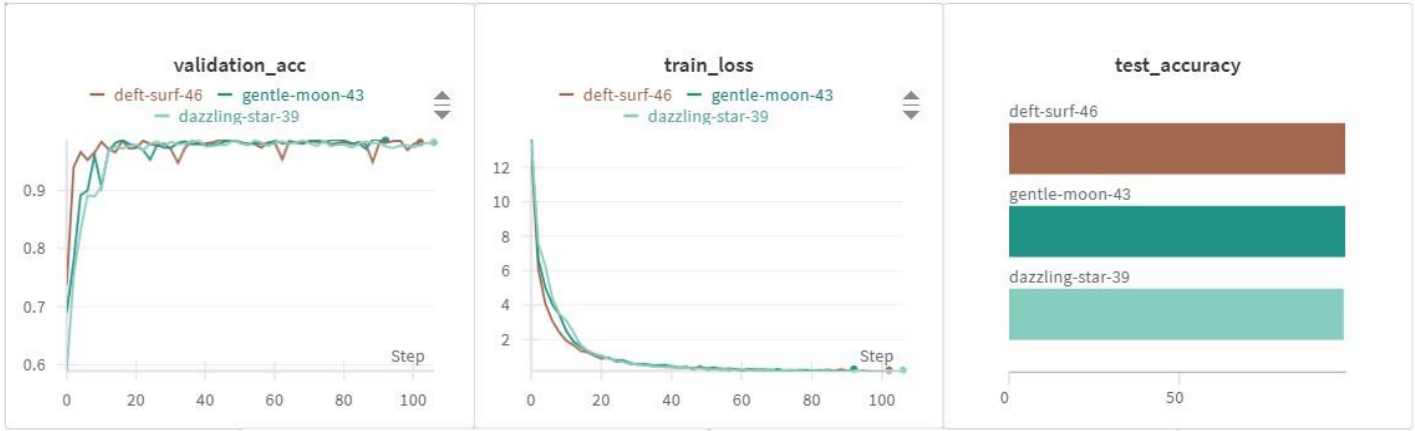
Име на изпълнение	Загуба	Тестова точност	Брой епохи	GPU памет (GB)
royal-voice-9	10.8631	45.27	150	0.71
dashing-monkey-7	11.0258	44.49	150	0.98
dazzling-paper-4	10.2014	45.75	150	0.98





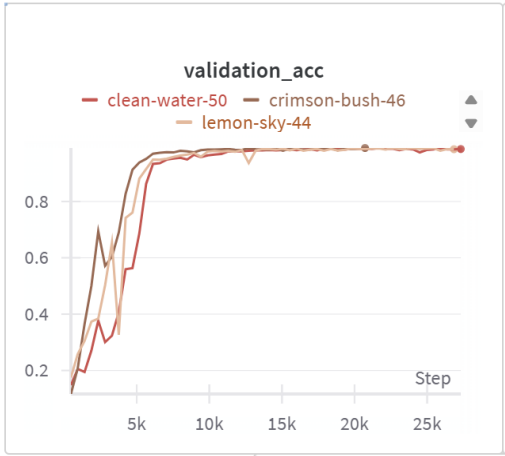
Експерименти - NoProp-DT и BLOODMNIST

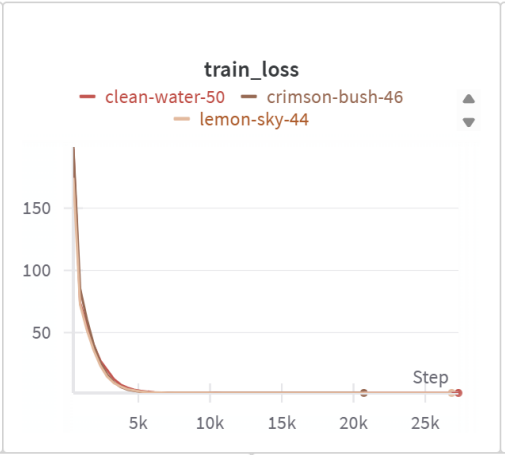
Име на изпълнение	Загуба	Тестова точност	Брой епохи	GPU памет (GB)
deft-surf-46	0.2152	98.25	52	4.50
gentle-moon-43	0.2713	98.42	47	4.71
dazzling-star-39	0.2163	97.98	54	4.71

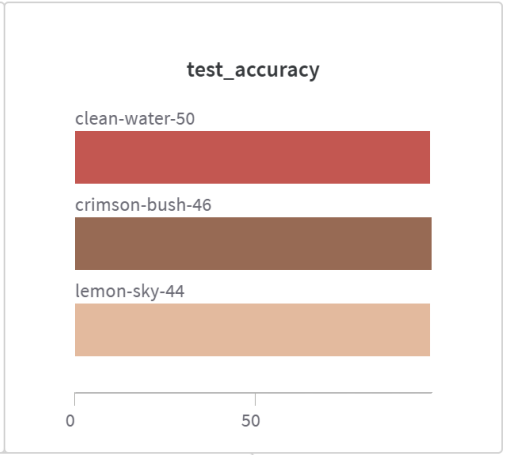


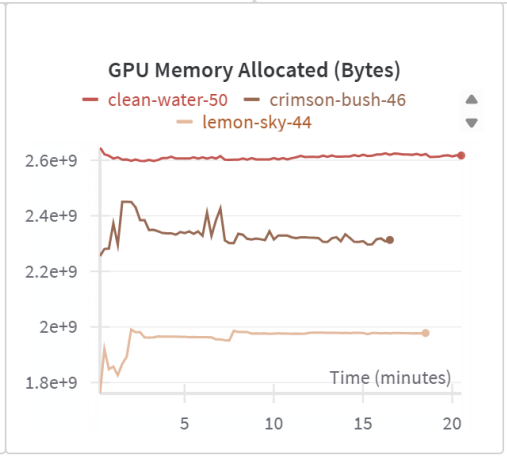
Експерименти - NoProp-CT и MNIST

Име на изпълнение	Загуба	Тестова точност	Брой епохи	GPU памет (GB)
clean-water-50	1.4221	98.63	58	2.65
crimson-bush-46	1.4950	99	44	2.45
lemon-sky-44	1.4934	98.57	57	1.99



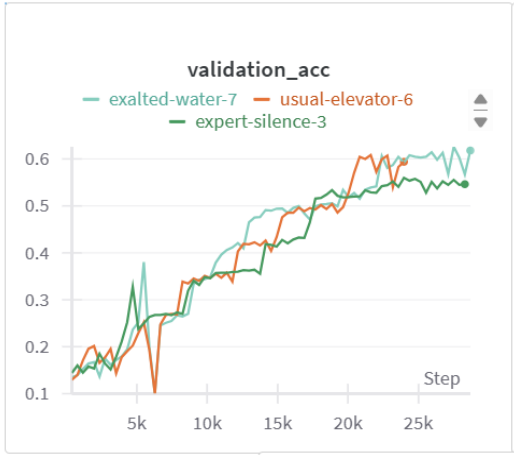


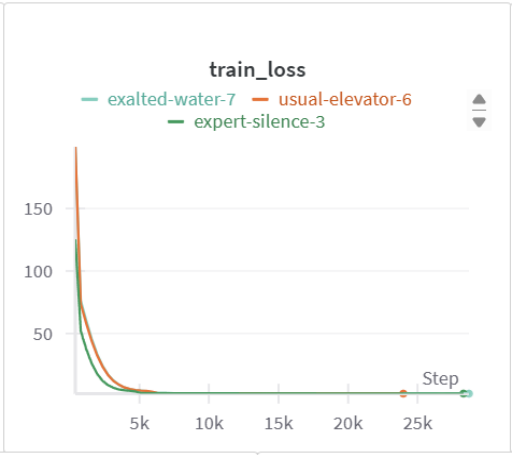


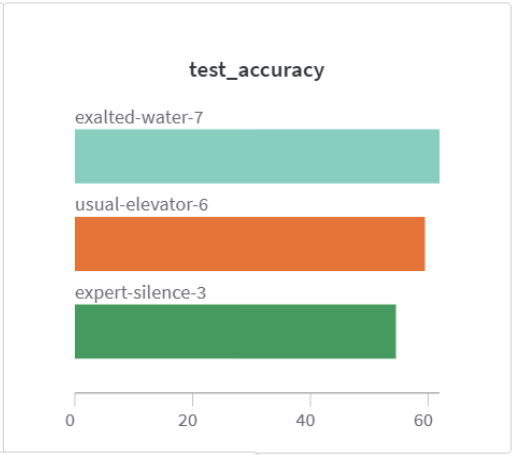


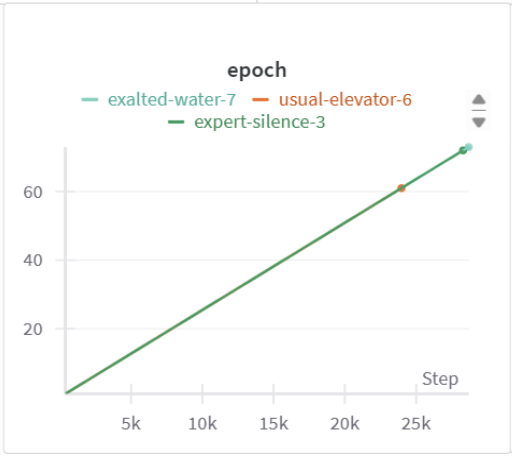
Експерименти - NoProp-CT и CIFAR-10

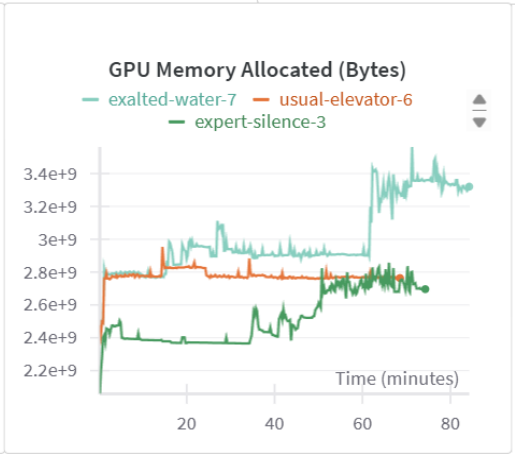
Име на изпълнение	Загуба	Тестова точност	Брой епохи	GPU памет (GB)
exalted-water-7	1.8715	61.79	73	3.17
usual-elevator-6	1.8880	59.39	61	2.75
expert-silence-3	1.9260	54.62	72	2.50





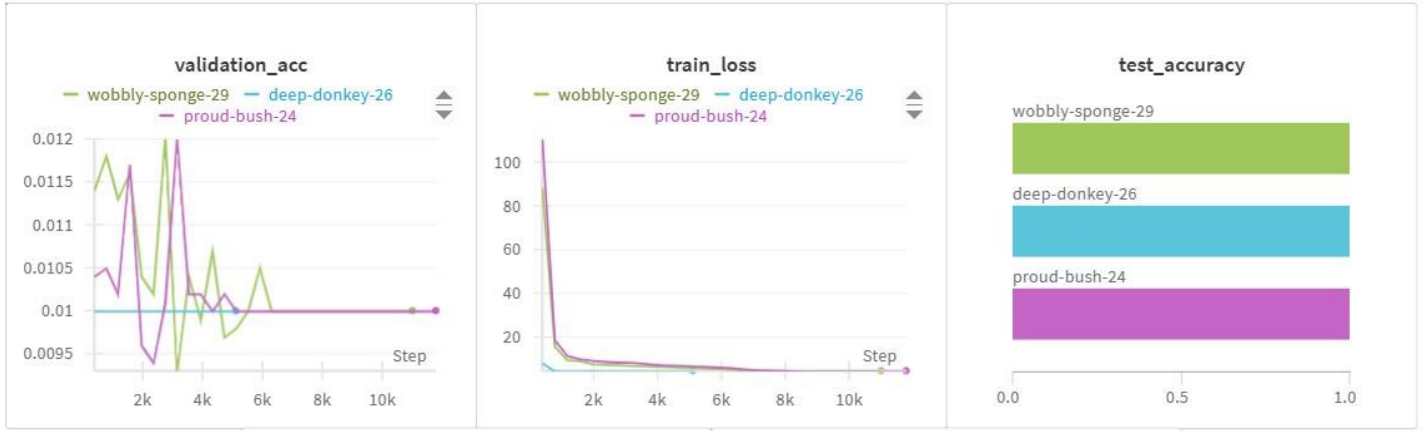






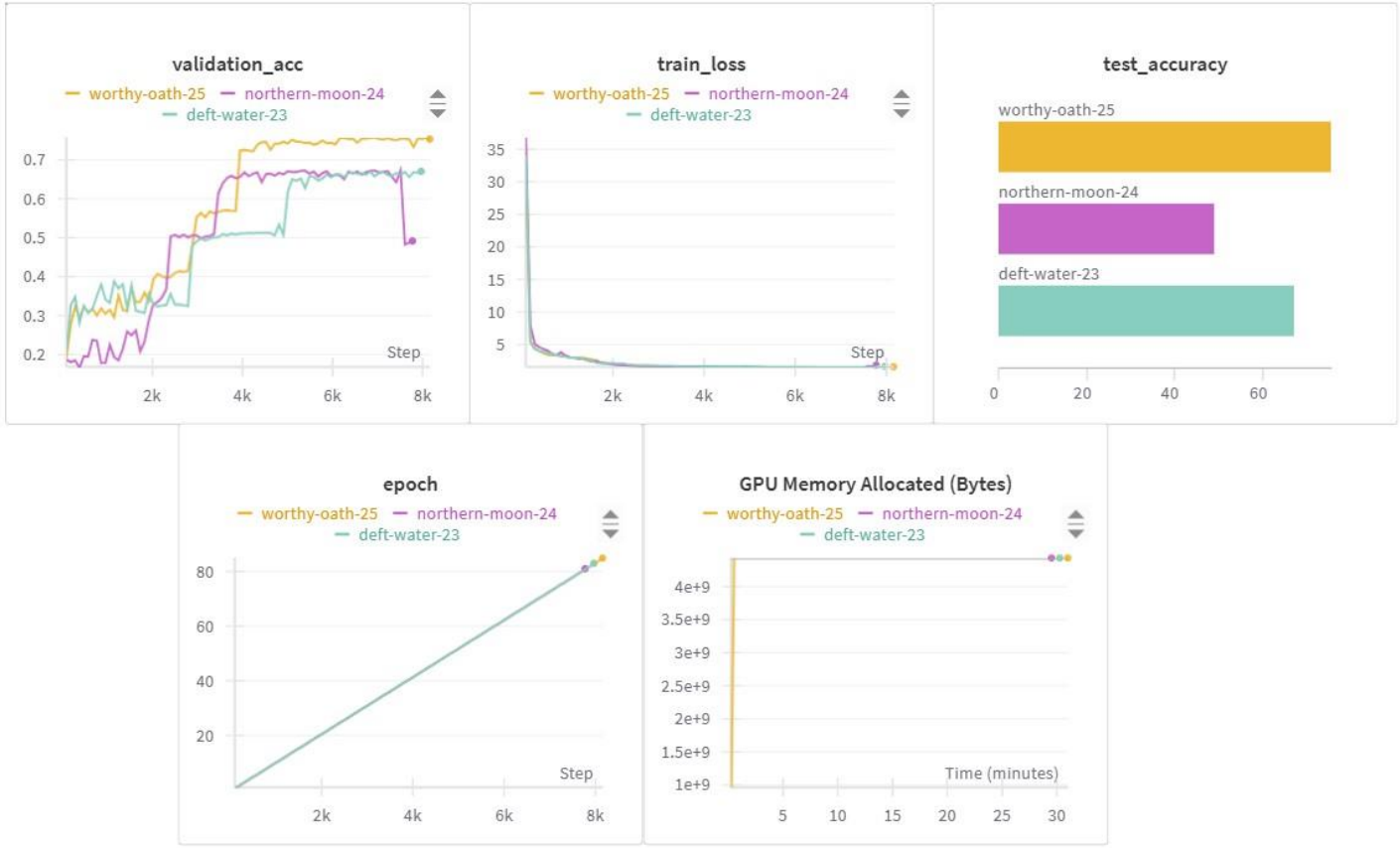
Експерименти - NoProp-CT и CIFAR-100

Име на изпълнение	Загуба	Тестова точност	Брой епохи	GPU памет (GB)
wobbly-sponge-29	4.6167	1	29	0.63
deep-donkey-26	4.6106	1	13	2.64
proud-bust-24	4.6209	1	30	1.91



Експерименти - NoProp-CT и BLOODMNIST

Име на изпълнение	Загуба	Тестова точност	Брой епохи	GPU памет (GB)
worthy-oath-25	1.5700	75.50	85	4.43
northern-moon-24	1.8103	49.08	81	4.43
deft-water-23	1.6466	67.14	83	4.43



Сравнение с оригиналните резултати

Dataset	Split	NoProp-CT		NoProp-DT	
		Paper	Implementation	Paper	Implementation
MNIST	Train	97.18 ± 1.02	$\mathbf{98.73 \pm 0.82}$	$\mathbf{99.97 \pm 0.0}$	99.44 ± 0.08
	Test	97.17 ± 0.94	$\mathbf{98.73 \pm 0.82}$	$\mathbf{99.54 \pm 0.04}$	99.44 ± 0.08
CIFAR-10	Train	$\mathbf{86.2 \pm 7.34}$	58.6 ± 3.65	$\mathbf{97.23 \pm 0.11}$	80.30 ± 0.6
	Test	$\mathbf{66.54 \pm 3.63}$	58.6 ± 3.65	$\mathbf{80.54 \pm 0.2}$	80.30 ± 0.6
CIFAR-100	Train	$\mathbf{40.88 \pm 10.72}$	0.23†	$\mathbf{90.7 \pm 0.14}$	45.17 ± 0.64
	Test	21.31 ± 4.17	**0.23†**	$\mathbf{46.06 \pm 0.25}$	45.17 ± 0.64
MEDMNIST	Train	-	$\mathbf{69.92 \pm 4.71}$	-	$\mathbf{98.39 \pm 0.24}$
	Test	-	$\mathbf{63.90 \pm 13.50}$	-	$\mathbf{98.22 \pm 0.22}$

† Тренирането на модела беше нестабилно

Сравнение на използваната памет

Вариант	MNIST	CIFAR-10	CIFAR-100
NoProp-DT	1.54 GB	1.15 GB	0.89 GB
NoProp-CT	2.36 GB	2.81 GB	1.73 GB

Сравнение на употребата на GPU RAM памет на имплементацията за различни методи и набори от данни.

Заключение: Постигнати резултати

- NoProp-DT постига висока точност на класификация върху наборите от данни MNIST, CIFAR-10, CIFAR-100 и BLOODMNIST
  - В някои случаи надхвърля или се доближава до оригиналните резултати от статията
- NoProp-CT показва стабилност върху MNIST и BLOODMNIST
- NoProp-CT показва по-ниска производителност върху по-сложни набори от данни като CIFAR-10 и CIFAR-100

Заключение: Изводи

- Предимства:
  - Възможността за паралелна обработка
  - Намалени изисквания за памет
  - Потенциал за ускоряване на обучението
  - Приложимост към изображения с по-висока резолюция
- Недостатъци:
  - Значителна сложност на имплементацията в сравнение с back-propagation
  - Повишена вероятност от бъгове и логически грешки
  - Относителна нестабилност на тренирането, която не се наблюдава при back-propagation
  - Голяма дисперсия и немонотонност в метриките на резултатите от тренирането