Architektury sieci neuronowych i ich uczenie w tensorflow

SZTUCZNA INTELIGENCJA I AUTOMATYZACJA PROCESÓW BIZNESOWYCH W UJĘCIU TECHNICZNYM

31.01.2026

© MGR INŻ. NATALIA POTRYKUS

Program

3 spotkania:

- 17.01.2026 r. 13:00-16:15
  Encoder-decoder, segmentacja obrazu w praktyce
- 31.01.2026 r. 9:00-12:15
  Dostrajanie hiperparametrów, praca z danymi sekwencyjnymi, generowanie tekstu
- 31.01.2026 r. 13:00-16:15
  Transformery: pierwsze starcie, konsultacje, Q&A

Dostrajanie hiperparametrów: praktyczny poradnik treningu

- Podejście systematyczne: grid search / random search na wybranych hiperparametrach
- Grid search: *n* hiperparametrów *h* do sprawdzenia, dla każdego z nich k_i opcji - ile potrzeba prób?

$$X = h_1k_1 * h_2k_2 * \dots * h_nk_n$$

- Robi się tego naprawdę sporo...
- Grid search - tak, ale nie dla wszystkich hiperparametrów
- Narzędzia do automatyzacji strojenia: GridSearchCV, RandomizedSearchCV (scikit-learn), KerasTuner, Optuna

Mam wybrany model - i co dalej?

Jeżeli nie wiesz, od czego zacząć, zacznij od domyślnych wartości gdzie się da. Gdzie się nie da, zacznij na oko*

*sprawdź parę przykładów na podobnym zbiorze / podobnym modelu.

Learning rate

“Size of the step down the loss function curve - Długość kroku w dół doliny straty...” (automatyczne tłumaczenia rządzą 💚)

Zwiększ, jeśli:	Zmniejsz, jeśli:
- train loss spada, ale bardzo powoli	- train loss skacze i szaleje - accuracy też jest niestabilne

W praktyce: lr = 0.001 jest częstym punktem startu

O ile ruszyć? Dopisz/usuń zero po kropce :)

Batch size

“Ile przykładów naraz pytam o kierunek treningu?”

Zwiększ, jeśli:	Zmniejsz, jeśli:
- loss mocno się waha - trening jest “za” wolny :)	- model szybko się przeucza - overfitting (train acc/loss wyglądają świetnie, val acc/loss - gorzej)

W praktyce: niewielkie potęgi dwójki (16-32) są częstym punktem startu

O ile ruszyć? Sprawdź kolejną potęgę dwójki.

Pamiętaj o ograniczeniach pamięci CPU/GPU - większy batch może się nie zmieścić, CUDA nas poinformuje stwierdzeniem “CUDA out of memory”. Wtedy nie pozostaje nam nic innego, jak trzymać się mniejszych wartości batch size.

Batch normalization

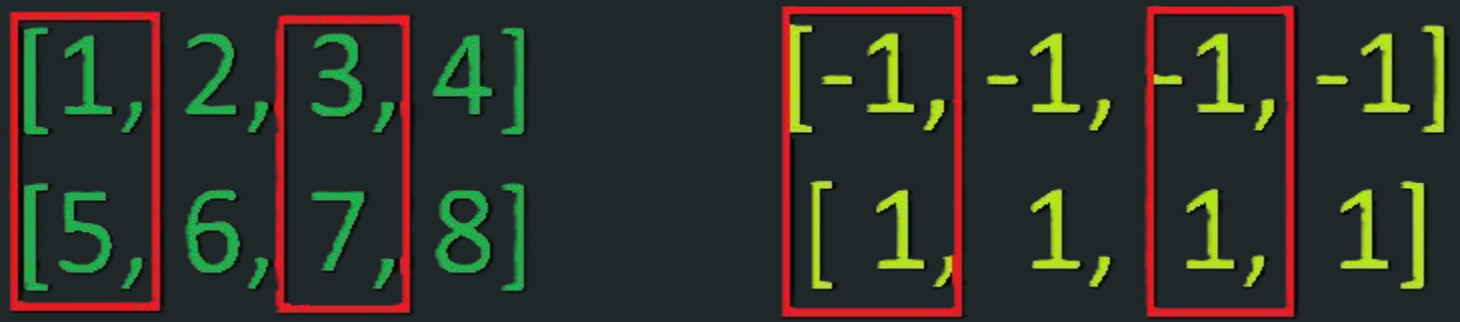
“Czy chcę modelowi dawać dane z znanym zakresie - znormalizowane?”

Spoiler: pewnie tak...

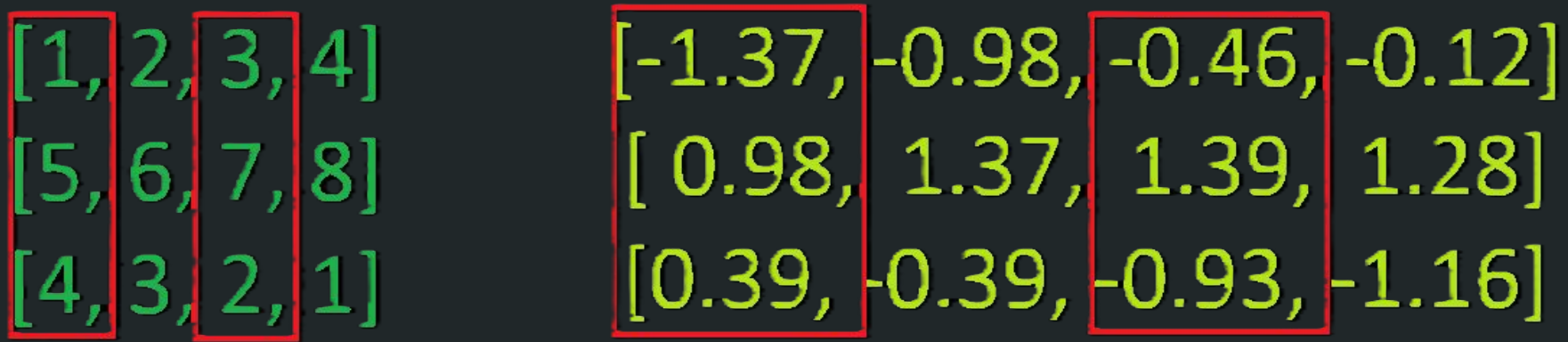
Dodaj, jeśli:	Usuń, jeśli:
- Przy treningu CNN/MLP, szczególnie, jeżeli accuracy i loss się wahają	- Batch size jest bardzo mały (1, 2 próbki)

Keras: BatchNormalization.

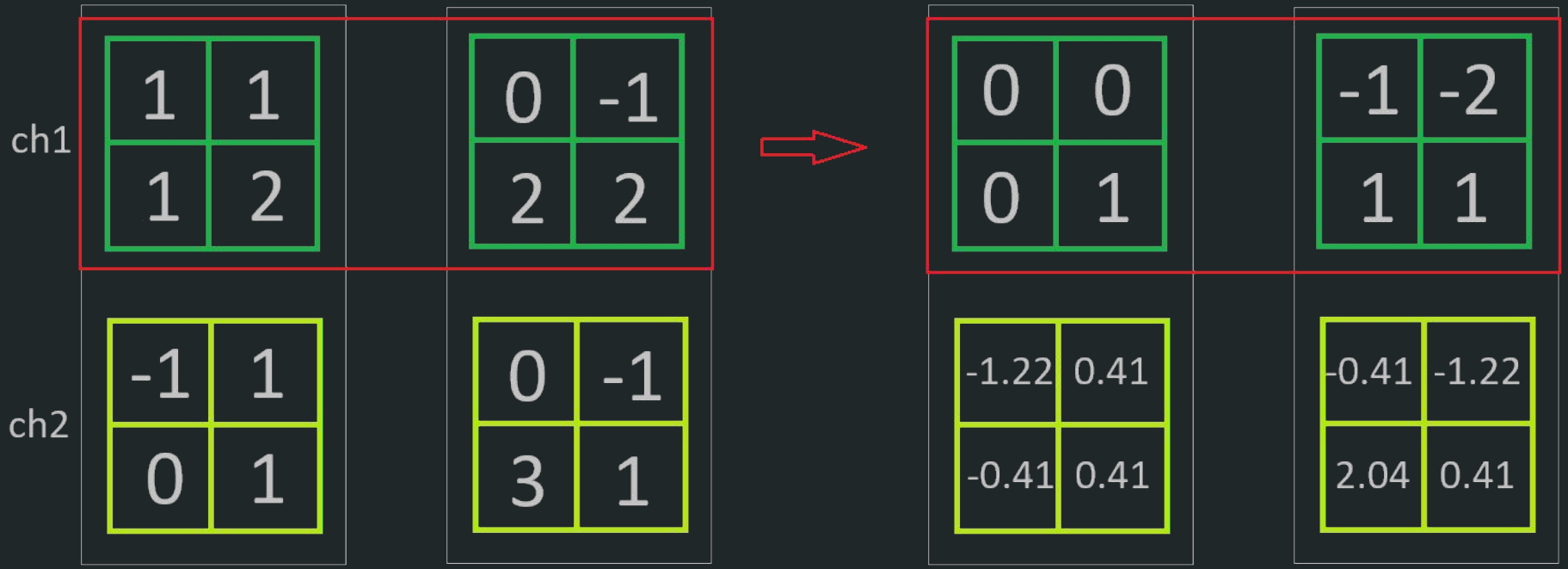
Batch normalization (1d)



Batch normalization (1d)



Batch normalization (2d)



Weight decay, dropout

“Jak wiernie model ma się sugerować wszystkim, co widzi podczas treningu”

Zwiększ (lub dodaj), jeśli:	Zmniejsz, jeśli:
- występuje overfitting	- model nie potrafi się uczyć (nawet wyniki treningowe są słabe) - Trening jest “za” wolny

W praktyce: dobrze zacząć od małego weight decay (np. wd = 0.0001). Jeżeli zdecydujemy się na warstwy dropout, najczęściej ich stopień (to, ile neuronów będzie “wyłączanych” podczas treningu) ustawia się na 0.1-0.5.

O ile ruszyć? O 0.1.

Model

“Czy mój model w ogóle stanie na wysokości zadania?”

Zwiększ, jeśli:	Zmniejsz, jeśli
- train/loss jest wysoki i nie chce spadać, mimo modyfikacji hiperparametrów - wyniki nie poprawiają się niezależnie od rozmiaru zbioru treningowego	- występuje overfitting mimo modyfikacji hiperparametrów

Inne sprawy

Optimizer: Zazwyczaj po prostu Adam, można próbować mini-batch gradient descent. Zmienia się, jeżeli modyfikacja LR (i innych hiperparametrów) niewiele daje (wtedy na przykład SGD, AdamW).

Seeding: dla powtarzalności wyników, frameworki umożliwiają *seedowanie* - ustawianie ziarna generatora liczb pseudolosowych

Augmentacje danych: jeżeli nasz zbiór jest mały / niezbalansowany, augmentacje mogą pomóc wyrównać liczebność klas oraz zwiększyć jego rozmiar.

EarlyStopping (patience): parametr pomagający skrócić czas treningu / zapobiegać przetrenowaniu. Ustala się liczbę epok i kryterium optymalizacyjne do przerwania treningu, np. *jeżeli przez 10 epok metryka val/loss się zmniejszy, pomiń pozostałe epoki treningu*.

Często zapisuje się nie tylko ostatni zestaw wag (stan modelu po całym treningu), ale i jeden (lub kilka) *checkpointów*, które wykazały najlepsze działanie podczas treningu względem kryterium optymalizacyjnego, (np. *train/acc*). Keras pozwala na łatwe zapisanie najlepszego (zamiast ostatniego) modelu w opcjach ModelCheckpoint: *save_best_only*.