Pomijamy noise, tj. błąd pomiaru g(x). Dobre interaktywne wyjaśnienie biorące pod uwagę noise: https://mlu-explain.github.io/bias-variance/

Bias variance trade-off

- rozwiązujemy problem regresji. Skończone zbiory treningowe D:(x_i, $g(x_i)$) stanowią niedoskonałe przybliżenie pewnej funkcji g(x)
- uczymy modele f(x) na podstawie różnych D
- średni (po wszystkich możliwych D) błąd kwad., który popełnimy dla konkretnego x wynosi:

$$\mathbb{E}_D[(f(x) - g(x))^2] = (\mathbb{E}_D[f(x)] - g(x))^2 + \mathbb{E}_D\left[(\mathbb{E}_D[f(x)] - f(x))^2\right]$$

Błąd kwadratowy dla konkretnego x

Bias: ile *średnie wskazanie* różni się od g(x) (“średnie wskazanie” to średnia wskazań wszystkich możliwych modeli)

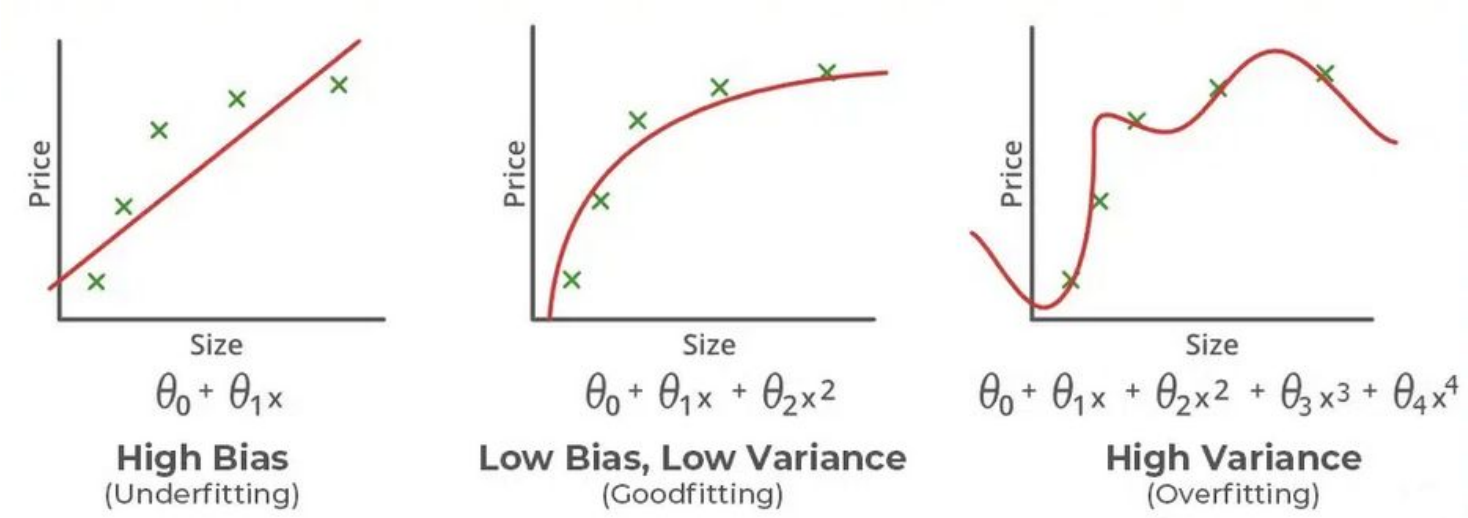
Wariancja: ile *średnie wskazanie* średnio różni się od konkretnego wskazania

Średnia po wszystkich możliwych zbiorach treningowych D

Dla innych fcji straty wychodzi inaczej, ale intuicja jest ta sama: błąd wynika z niedokładności (bias) i zmienności (wariancja) modelu. Model powinien popełniać mały błąd gdy uśrednimy go po wszystkich możliwych zbiorach treningowych i nie miotać się bardzo dla poszczególnych zbiorów treningowych.

Overfitting

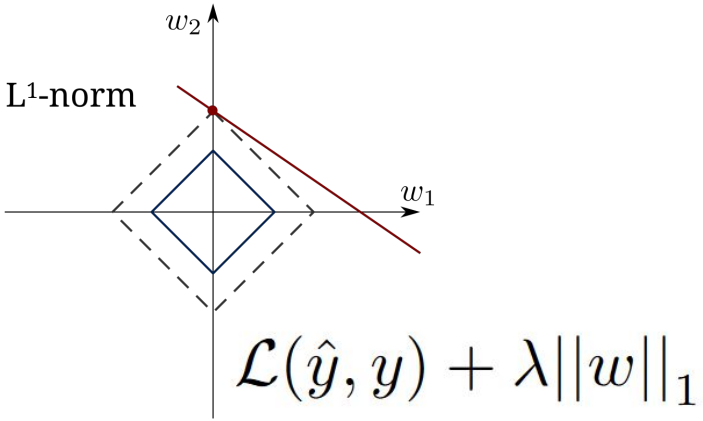
- model mocno dopasowuje się do danych treningowych i zaczyna działać gorzej na danych testowych
- występuje wysoka wariancja, czyli gwałtowna zmiana modelu przy zmianie zbioru treningowego

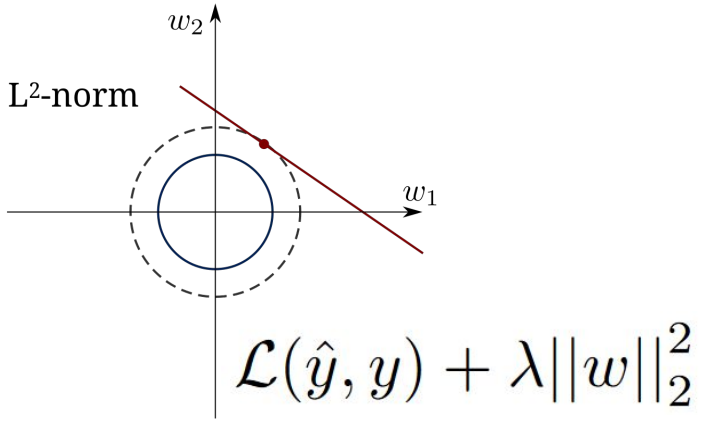


https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

Regularyzacja

- zestaw technik zapobiegających overfittingowi (“upraszczających” model)
- najprostszy sposób: do funkcji straty dodać karę uzależnioną od parametrów
- regularyzacja L1: sum(abs(parametry))
- regularyzacja L2: sum(parametry^2)
- w prostych modelach (np. w regresji liniowej) L1 prowadzi do zerowania części parametrów, zaś L2 do preferowania mniejszych parametrów
- można pokazać, że w regresji liniowej z błędem kwadr., L2 zmniejsza wariancję

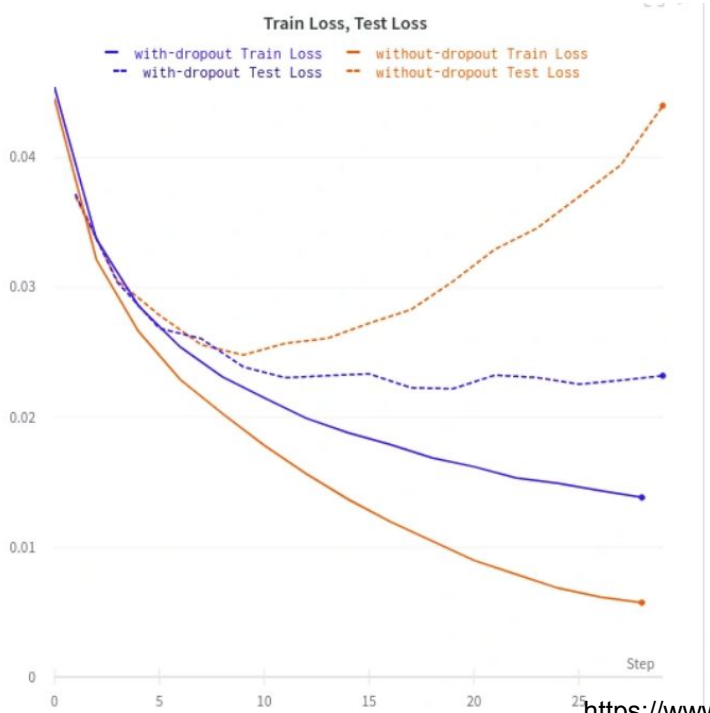


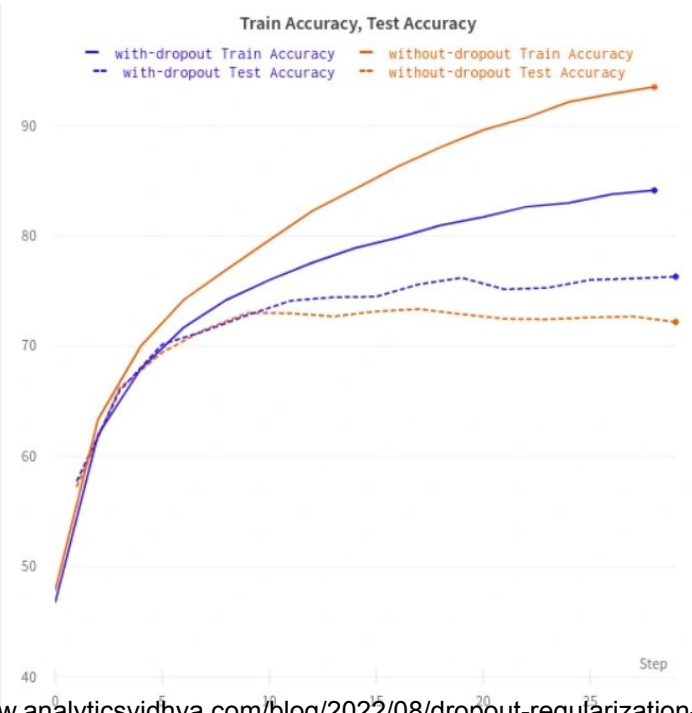


https://commons.wikimedia.org/w/index.php?curid=58258966

Overfitting sieci głębokich

- widoczny na wykresie validation loss



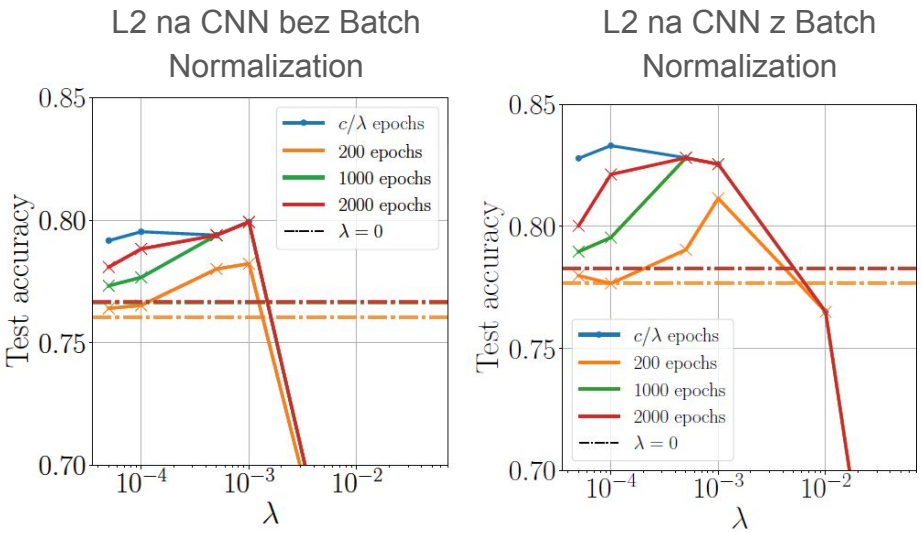


https://www.analyticsvidhya.com/blog/2022/08/dropout-regularization-in-deep-learning/

Regularyzacja sieci głębokich

- sieci nie zachowują się jak proste modele
- powszechnie znane regularyzatory sieci głębokich: data augmentation, early stopping, dropout, weight decay
- podejrzewane o regularyzację: SGD, batch normalization, dosłownie wszystkie inne w zal. od źródła
- SGD + reg. L2 = SGD + weight decay
- Adam + reg. L2 ≠ Adam + weight decay (weight decay działa lepiej)

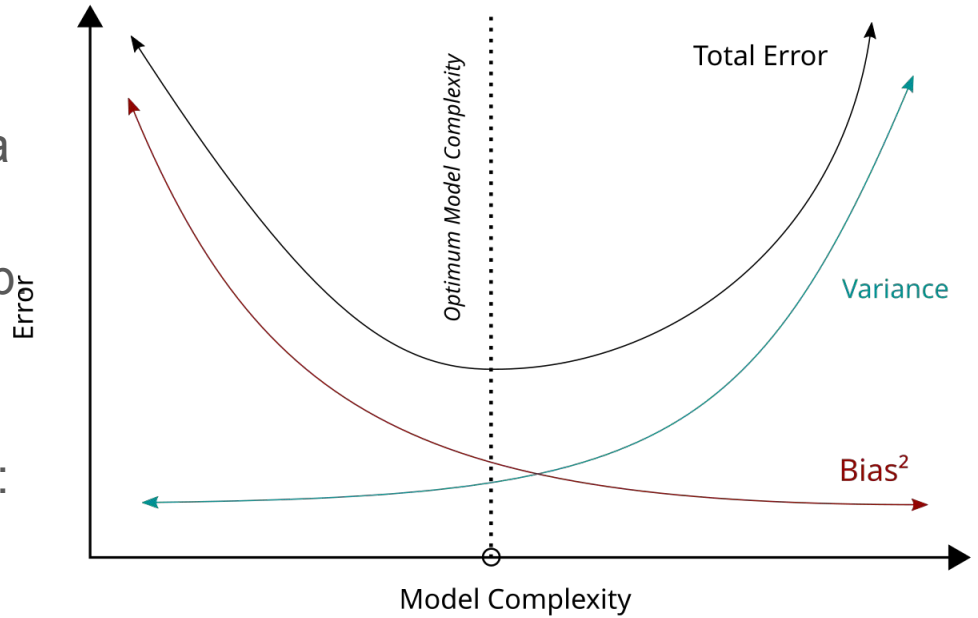
- regularyzatory mają złożony wpływ na wyniki sieci i wchodzą w interakcje z innymi elementami



On the training dynamics of deep networks with L2 regularization (2020)

Overfitting a liczba parametrów modelu

- według tradycyjnej teorii overfitting wynika z nadmiernego skomplikowania modelu
- skomplikowanie można mierzyć na wiele sposobów, jednak nie jest to po prostu liczba parametrów, bo np f(x) = a sin(bx)
- uczenie sieci można przerwać zanim pojawi się mocny overfitting: czy jest wtedy mniej skomplikowana?



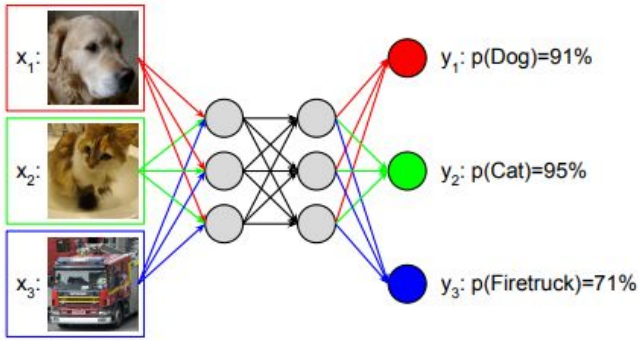
https://commons.wikimedia.org/w/index.php?curid=105307219

Sieci głębokie mają niezwykle dużo parametrów

- …a mimo tego naturalnie się regularyzują (zapewne dzięki SGD)
- sieć o 100k parametrów potrafi nauczyć się na 1k elementach
- przykładowa miara złożoności sieci: *Intrinsic Dimension* - jak wiele losowych kierunków w przestrzeni parametrów wystarcza, by dopasować model do danego zbioru treningowego
- prostszy problem -> niższe ID
- dla prostszych problemów w jednej sieci można zmieścić kilka naraz

Dataset	MNIST		MNIST (Shuf Pixels)		MNIST (Shuf Labels)
Network Type	FC	LeNet	FC	LeNet	FC
Parameter Dim. D	199,210	44,426	199,210	44,426	959,610
Intrinsic Dim. d_{int90}	750	290	750	1,400	190,000

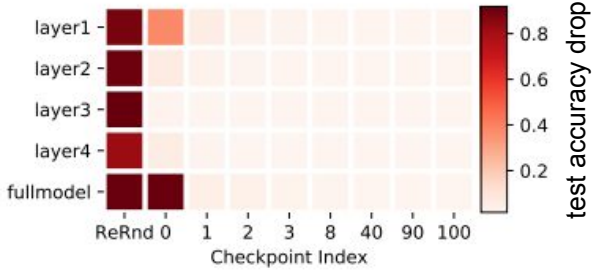
...	CIFAR-10		ImageNet	Inverted Pendulum	Humanoid	Atari Pong
...	FC	LeNet	SqueezeNet	FC	FC	ConvNet
...	656,810	62,006	1,248,424	562	166,673	1,005,974
...	9,000	2,900	> 500k	4	700	6,000



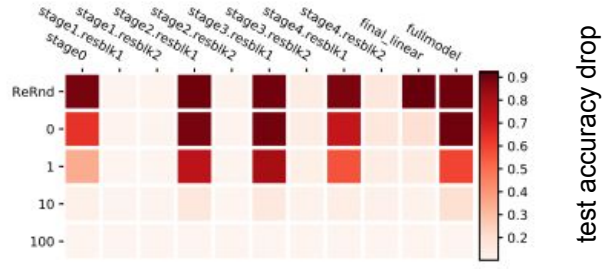
https://lilianweng.github.io/posts/2019-03-14-overfit/
Training Independent Subnetworks for Robust Prediction (2021)
Measuring the Intrinsic Dimension of Objective Landscapes (2018)
Reconciling modern machine learning practice and the bias-variance trade-off (2018)

Sieci nie wykorzystują wszystkich swoich parametrów

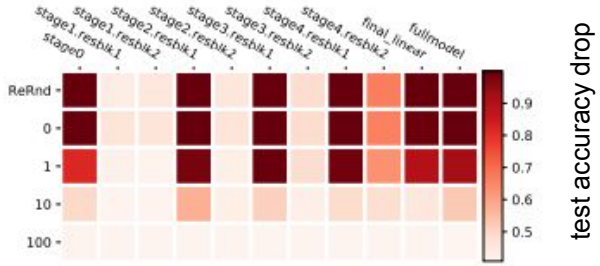
- zwykle ID < liczba parametrów. Czasem da się znaleźć całe niewykorzystane warstwy
- *reinitialization robustness*: cofamy wybraną warstwę w przeszłość (np. do 1. epoki)
- trening niektórych warstw ma małe znaczenie
- trudny problem ≈ więcej użytecznych warstw



A small fully-connected network on MNIST



ResNet on CIFAR: residual blocks spread complexity



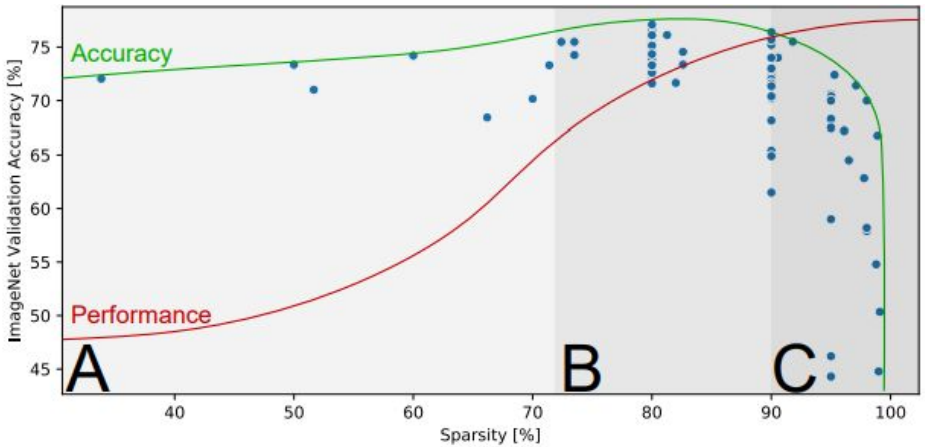
ResNet on ImageNet: more layers become critical

Are All Layers Created Equal? (2019)

Rzadkie sieci

- niektóre fragmenty sieci można usunąć bez pogorszenia wyników
- pozwala to oszczędzić czas i pamięć, a także może prowadzić do lepszej generalizacji
- pruning sieci można przeprowadzać podczas uczenia lub po jego zakończeniu
- choć teoria na temat pruningu jest rozbudowana, praktycznych przykładów użycia jest niewiele

- inne sposoby na przyspieszenie sieci są obecnie bardziej popularne, np. obniżanie precyzji obliczeń



Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks (2021)

Sieci konwolucyjne są rzadkie

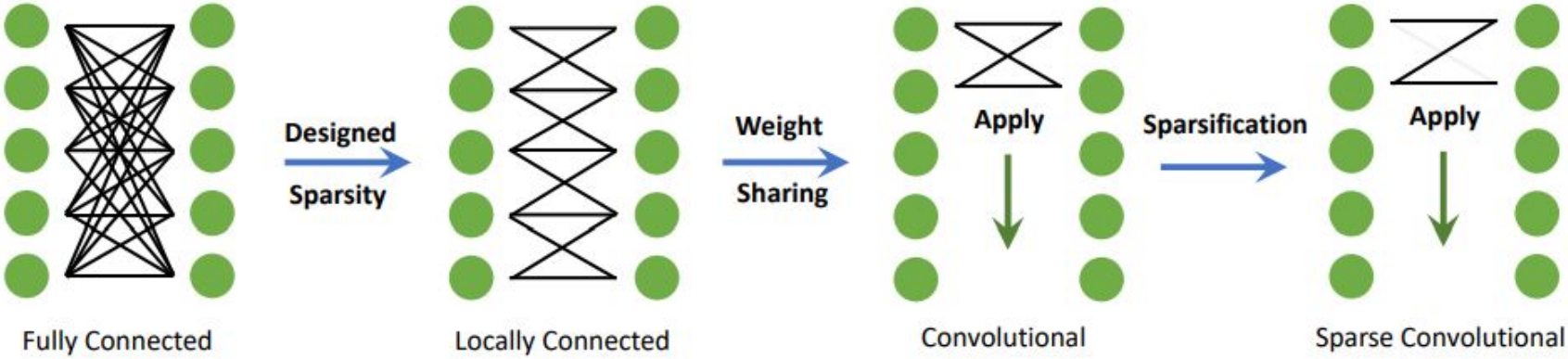
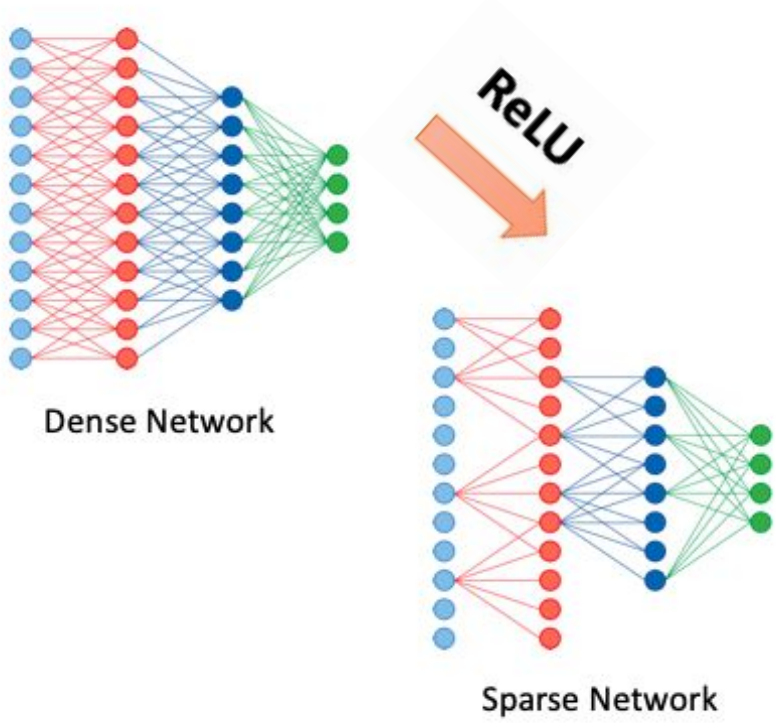


Fig. 3. Convolutional operators as sparse fully-connected operators for a single input and output channel.

Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks (2021)

ReLU tworzy rzadkie sieci

- neurony o wyjściu zero są w rzeczywistości wyłączane z sieci
- wyzerowane neurony nie uczą się na danym elemencie, bo nie przepływa przez nie gradient
- duża część sieci może być nieaktywna dla pojedynczych elementów
- być może pomaga to w generalizacji
- rzadką sytuację, w której wiele neuronów wyłącza się dla wszystkich elementów nazywamy “dying ReLU”

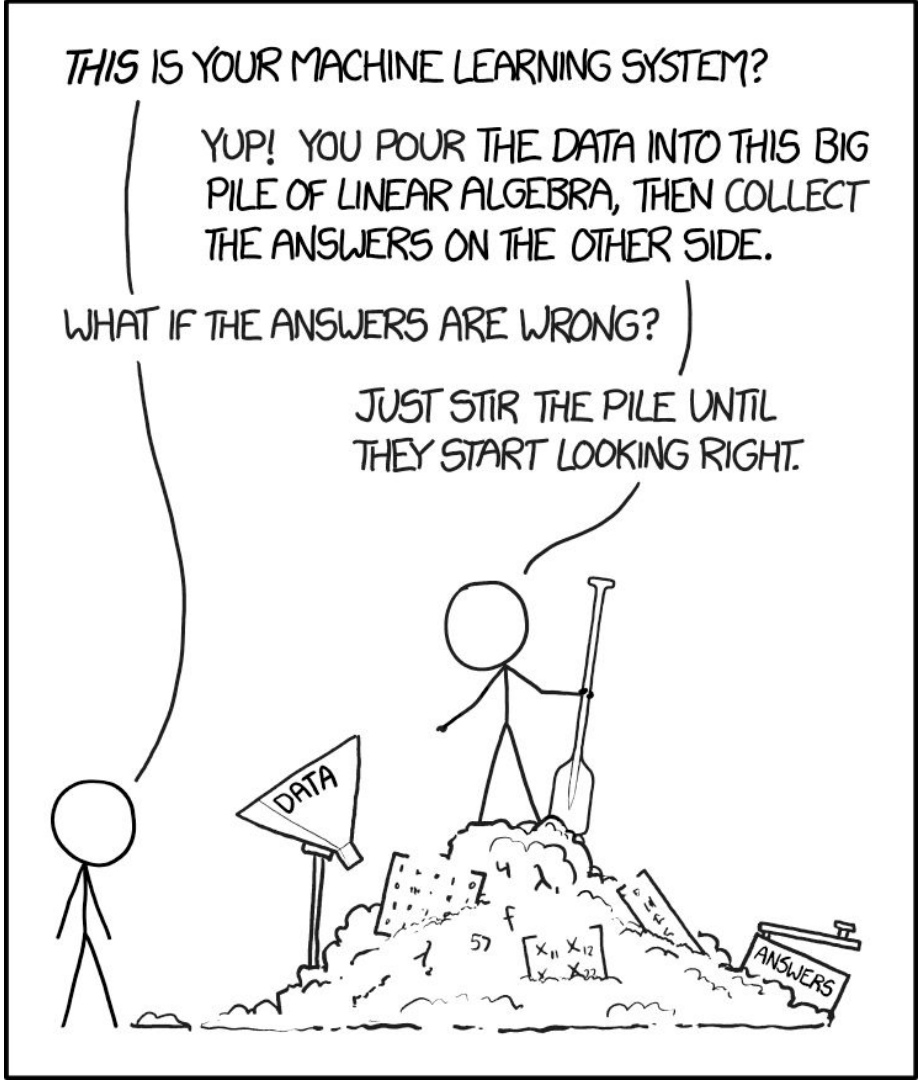


https://cerebras.ai/chip/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/

Lottery ticket hypothesis

- pruning pozwala zredukować niektóre sieci do 10% oryginalnej wielkości
- redukcja następuje po uczeniu, bo zredukowane sieci zwykle nie nauczyłyby się od zera
- a jednak, odkryto że sieci fully-connected zawierają podsieci ("winning tickets") które, gdy wyizolowane i uczone od zera, osiągają podobne test accuracy w tej samej liczbie iteracji treningowych
- ciekawe ze względów teoretycznych: sieć jest jak worek losowych inicjalizacji, z których tylko niewielka część (~10%) okazuje się przydatna
- praktyczne zastosowania są ograniczone: autorzy osiągnęli dobre wyniki dla prostych problemów i płytkich sieci. W dodatku winning tickets wykrywa się na koniec (ew. podczas) trenowania i nie ma gwarancji, że wyniki będą dobre

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks (2019)



https://xkcd.com/1838/

Double descent

- małe i średnie sieci kończą trening na fazie overfittingu, po której wynik na zbiorze testowym stale się pogarsza. Wystarczająco duże sieci zdają się nie mieć takiego problemu: ich straty na train i test maleją asymptotycznie zarówno w miarę wydłużania treningu, jak i w miarę powiększania modelu

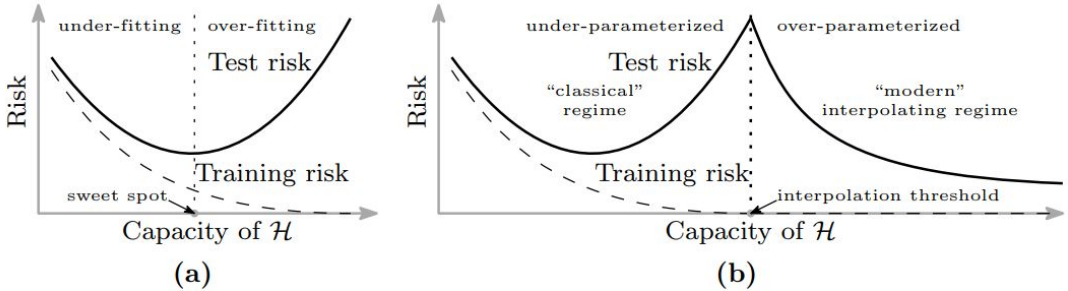


Figure 1: **Curves for training risk (dashed line) and test risk (solid line).** (**a**) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (**b**) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Reconciling modern machine learning practice and the bias-variance trade-off (2018)

Scaling law: wielkość modelu przy zadanym budżecie

- gdy nie grozi nam overfitting, jedynym ograniczeniem staje się budżet
- chcielibyśmy mieć nieskończenie duży model i uczyć go nieskończenie długo
- przy określonym budżecie większy model ≈ mniej uczenia
- dla ustalonej rodziny modeli optymalna wielkość modelu zdaje się spełniać prostą statystyczną zależność
- dla Llamy3 dobra_wielkość_train = 0.29 budżet 0.53

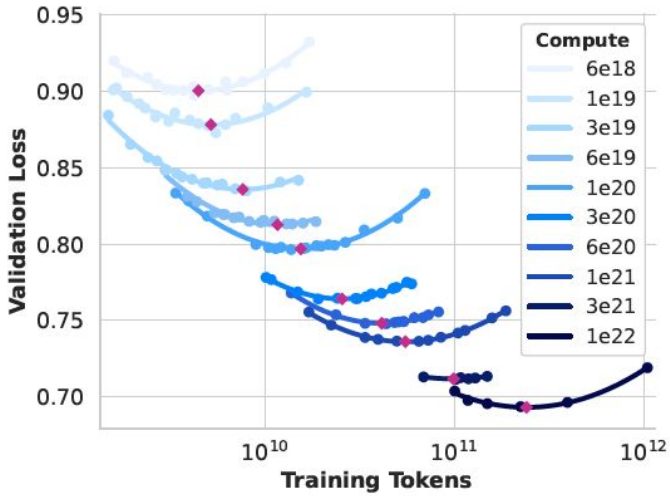
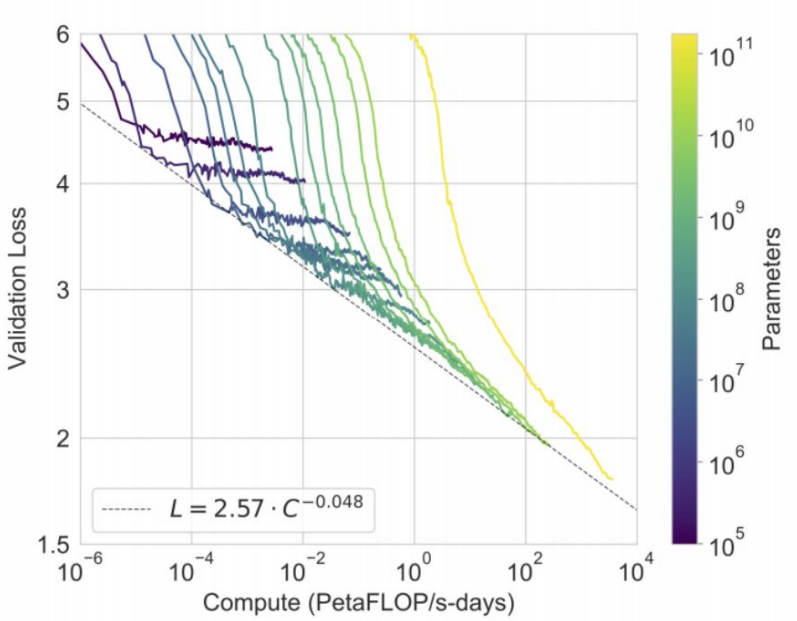


Figure 2 Scaling law IsoFLOPs curves between 6×10^{18} and 10^{22} FLOPs. The loss is the negative log-likelihood on a held-out validation set. We approximate measurements at each compute scale using a second degree polynomial.

The Llama 3 Herd of Models (2024)

Scaling law: przewidywanie wyników modelu

- ogólna postać *scaling law* to empiryczna zależność pomiędzy 4-ma wielkościami:
- N: parametry modelu
- D: elementy widziane podczas treningu
- C: koszt uczenia (budżet), C ~ N * D
- L: wynik sieci (strata na teście)
- zwykle bierzemy kilka małych C, tworzymy modele z naszej rodziny dla różnych (N, D, C), aby ustalić jak najlepsze (D, N) zależy od C (poprzedni slajd), a także oszacować, jak L zależy od C (po prawej)



- dzięki temu wiemy, jak dobrać (D, N) dla docelowego C i jakie L dostaniemy

Scaling law: przewidywanie wyników modelu

- scaling law pozwoliło przewidzieć stratę i wynik na benchmarkach Llama3

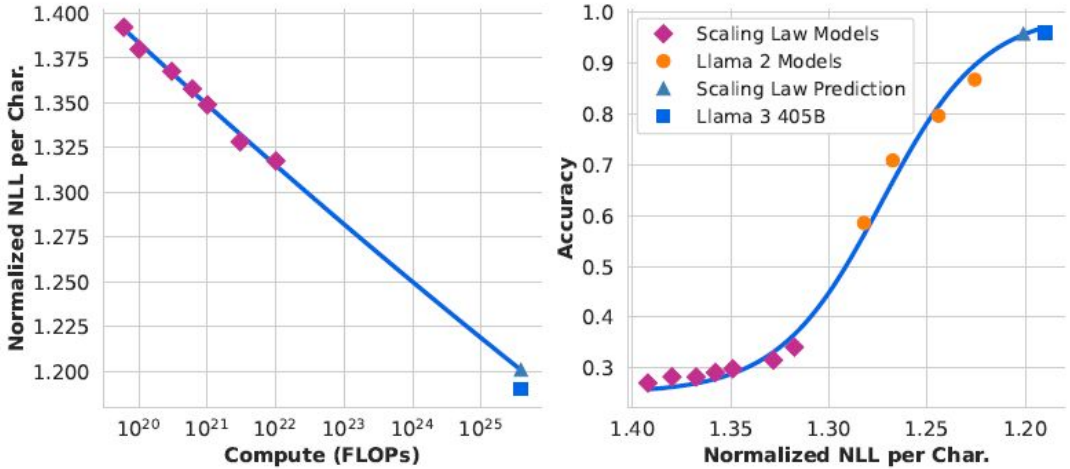


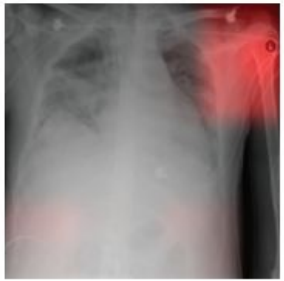
Figure 4 Scaling law forecast for ARC Challenge. *Left:* Normalized negative log-likelihood of the correct answer on the ARC Challenge benchmark as a function of pre-training FLOPs. *Right:* ARC Challenge benchmark accuracy as a function of the normalized negative log-likelihood of the correct answer. This analysis enables us to predict model performance on the ARC Challenge benchmark before pre-training commences. See text for details.

The Llama 3 Herd of Models (2024)

Shortcut learning in deep neural networks

Robert Geirhos ✉, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge & Felix A. Wichmann

Nature Machine Intelligence **2**, 665–673 (2020) | Cite this article

				**Article:** Super Bowl 50 **Paragraph:** "*Peython Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had a jersey number 37 in Champ Bowl XXXIV.*" **Question:** "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*" **Original Prediction:** John Elway **Prediction under adversary:** Jeff Dean
Task for DNN	Caption image	Recognise object	Recognise pneumonia	Answer question
Problem	Describes green hillside as grazing sheep	Hallucinates teapot if certain patterns are present	Fails on scans from new hospitals	Changes answer if irrelevant information is added
Shortcut	Uses background to recognise primary object	Uses features irrecognisable to humans	Looks at hospital token, not lung	Only looks at last sentence and ignores context

https://www.nature.com/articles/s42256-020-00257-z

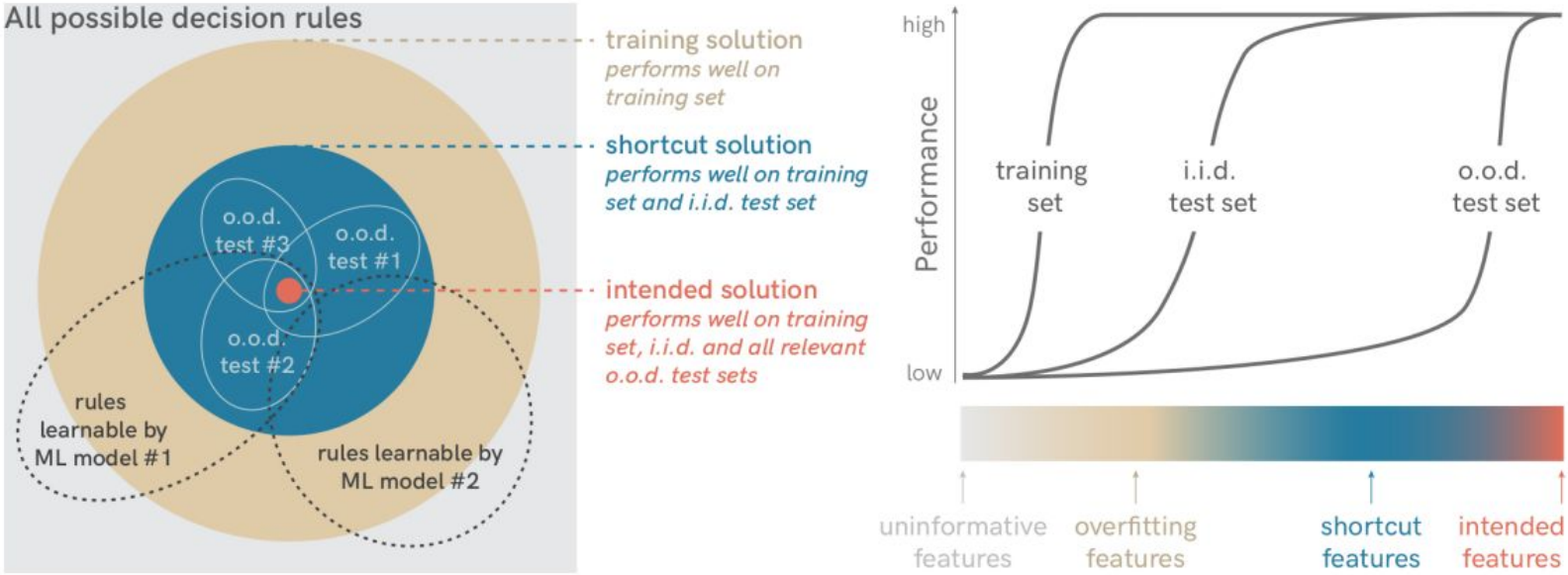


Figure 3. Taxonomy of decision rules. Among the set of all possible rules, only some solve the training data. Among the solutions that solve the training data, only some generalise to an i.i.d. test set. Among those solutions, shortcuts fail to generalise to different data (o.o.d. test sets), but the intended solution does generalise.

https://www.nature.com/articles/s42256-020-00257-z

Figure 4. Both human and machine vision generalise, but they generalise very differently. Left: image pairs that belong to the same category for humans, but not for DNNs. Right: images pairs assigned to the same category by a variety of DNNs, but not by humans.

https://www.nature.com/articles/s42256-020-00257-z

Data-centric AI

- buzzword który zwraca uwagę na ważne zjawisko: zwykle znacznie więcej czasu idzie na zbieranie i przygotowywanie danych niż na wybór i uczenie modelu
- teoria ML koncentruje się na modelach
- czasem większą poprawę można uzyskać pracując nad danymi

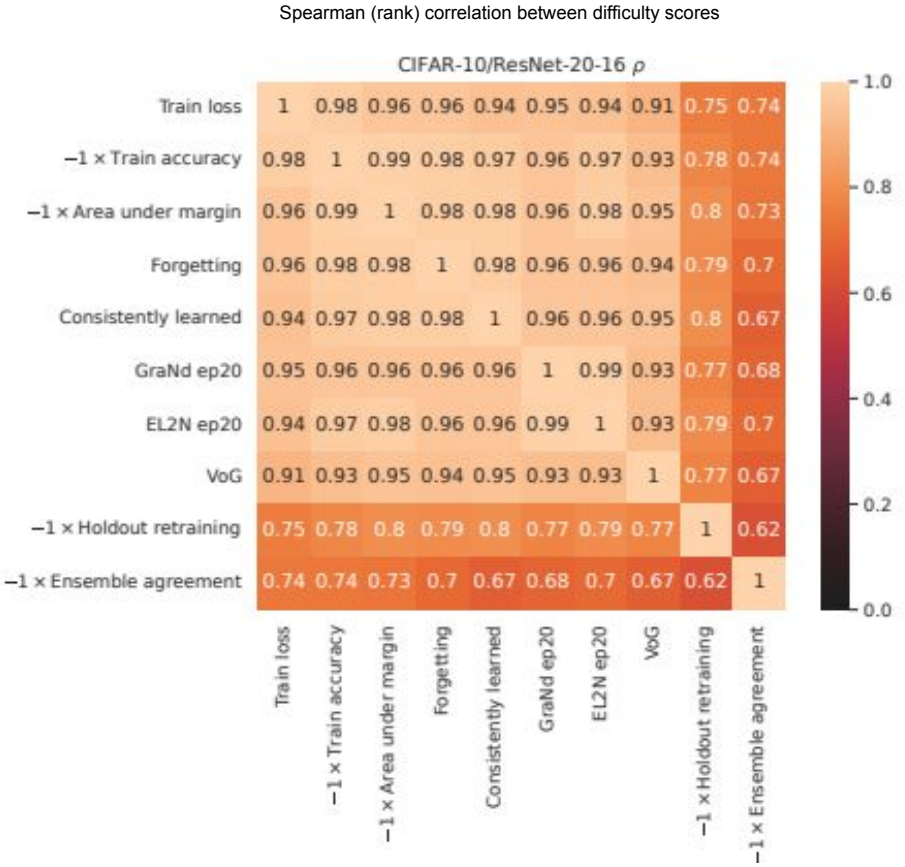
- *data-centric approach* to zwykle usuwanie outlierów, wykrywanie *mislabeled samples*, wspieranie zbierania danych (*active training*), *data augmentation* i *EDA*

CIFAR-10 given label:
cat

https://labelerrors.com/

Mierzenie trudności

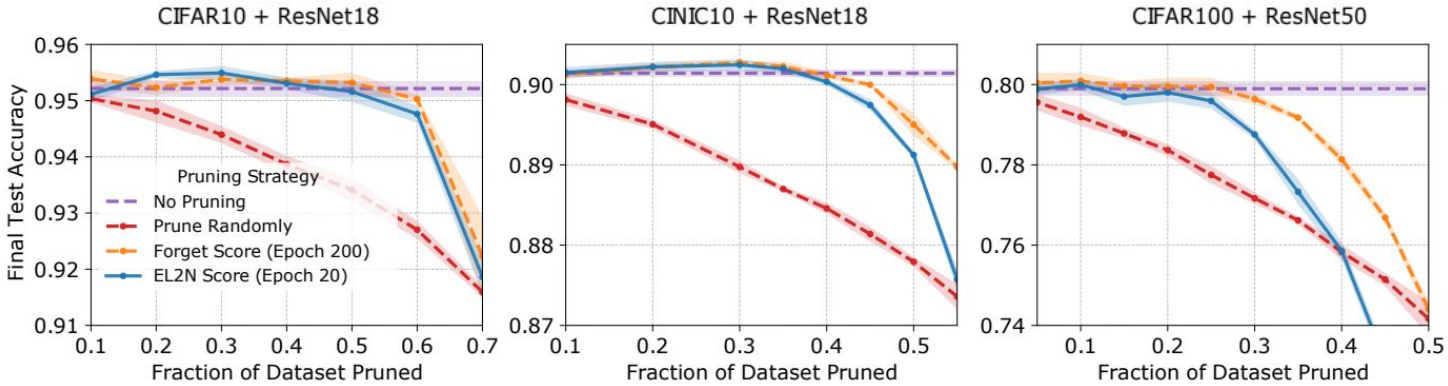
- elementy ze zbioru danych można uporządkować pod względem “trudności” dla danego modelu
- istnieje wiele miar trudności, w większości mocno skorelowanych
- niektóre wymagają, aby element należał do zbioru treningowego, a niektóre nie
- niestety, miary trudności są zwykle wrażliwe na losową inicjalizację; aby dokładnie je określić musimy nauczyć model wiele razy



Dataset Difficulty and the Role of Inductive Bias (2024)

Najłatwiejsze elementy są najmniej ważne

- balansowanie zbioru danych to nie tylko równanie klas: niektóre elementy są tak łatwe lub tak mylące, że usunięcie ich może prowadzić do lepszych wyników
- poniżej, do usunięcia najłatwiejszych el. zostały użyte dwie miary trudności: ile razy dany element został zapomniany podczas trenowania i jak mocno wpływał na gradient funkcji straty



Deep Learning on a Data Diet: Finding Important Examples Early in Training (2023)

Różne architektury mają trudności z różnymi elementami

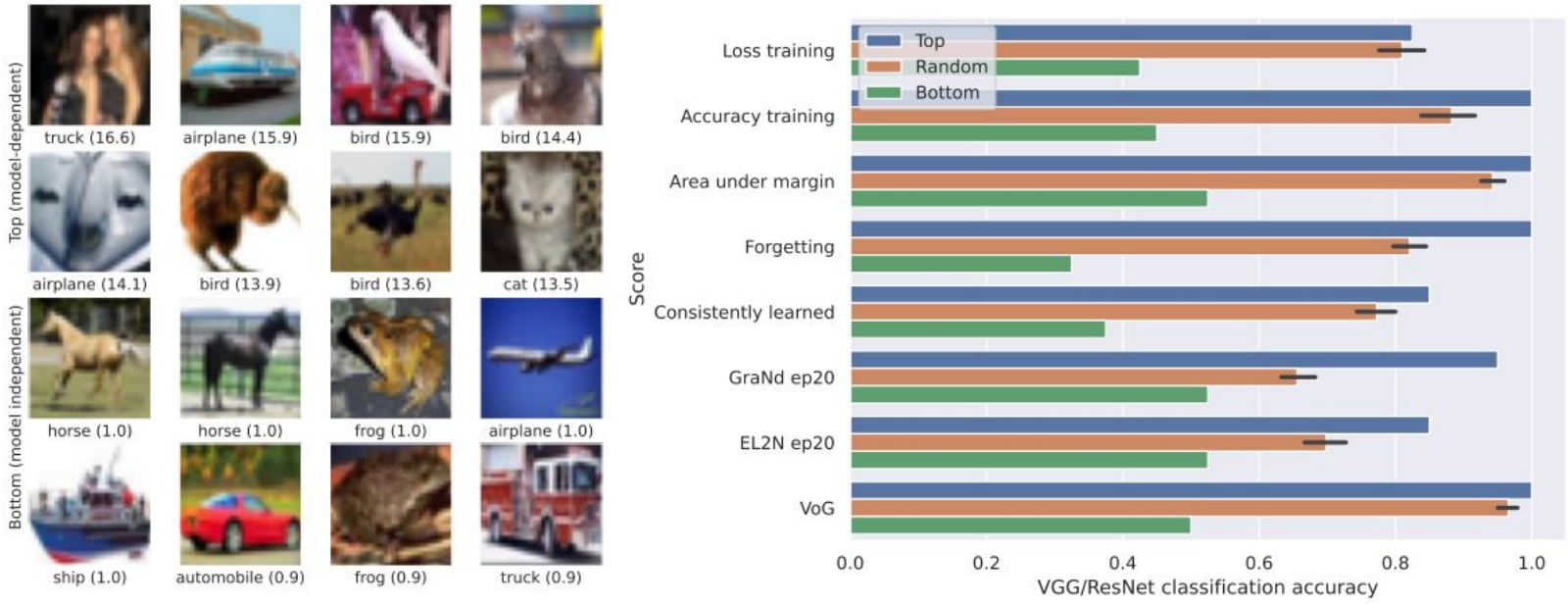
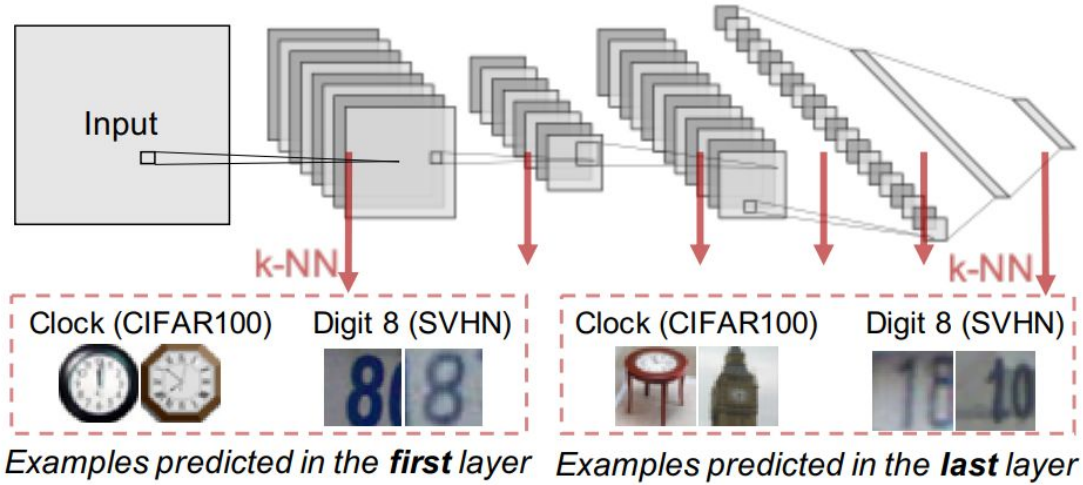


Figure 1: **Left:** Top 8 CIFAR-10 examples most/least sensitive to inductive biases (model width, depth, and architecture), ranked by statistical significance of changes to difficulty score for pairwise model comparisons (see Section 4.3 for more details). Numbers indicate mean $-\log P$ value of each example. **Right:** Distinguishing between VGG and ResNet using the top 8 examples (Left) as features for logistic regression.

Przykładowa miara trudności: prediction depth



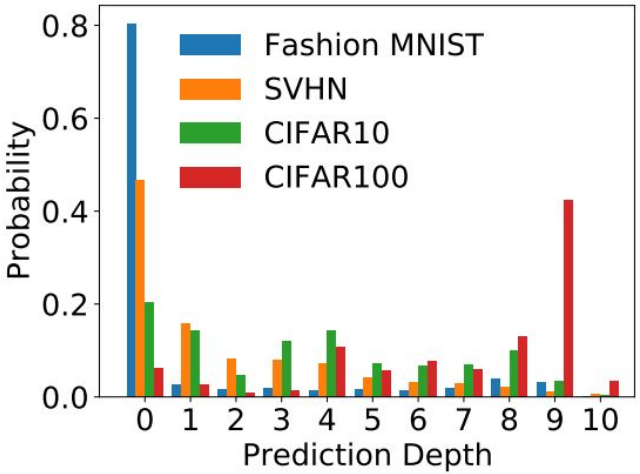


Figure 1: *Deep models use fewer layers to (effectively) determine the prediction for easy examples and more layers for hard examples.* **Left:** A cartoon illustrating the definition of prediction depth (given in Section 2.1).

Deep Learning Through the Lens of Example Difficulty (2021)

Dzielenie danych za pomocą prediction depth

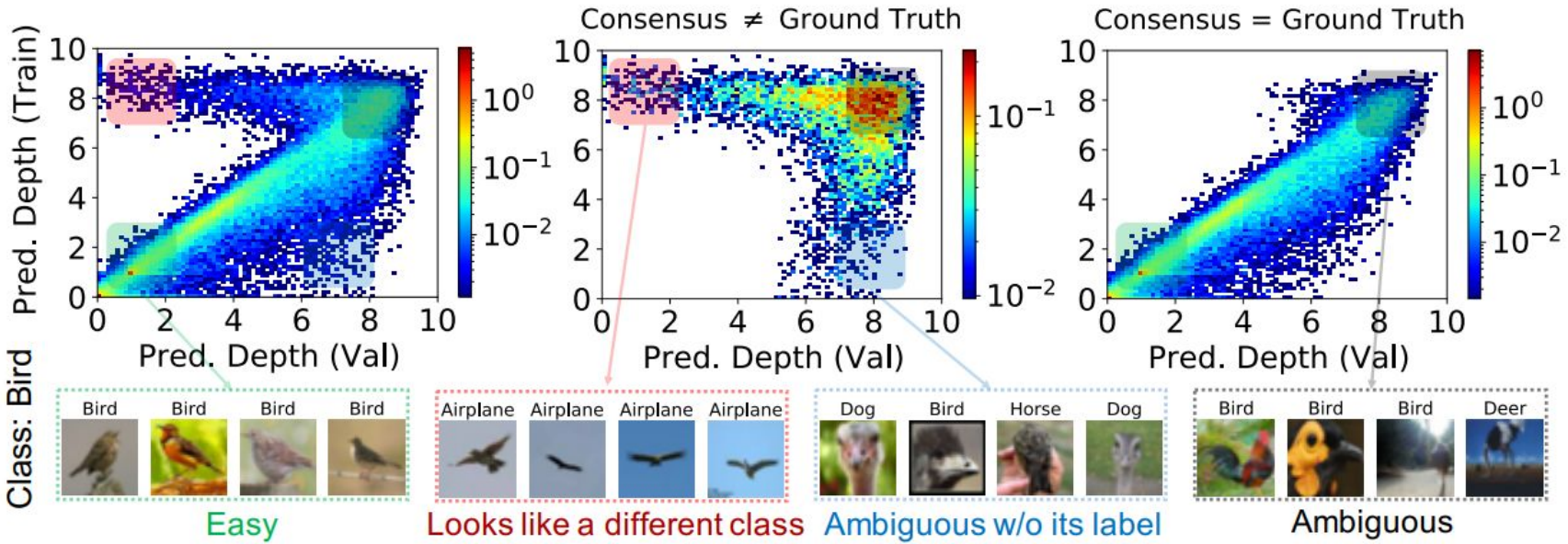


Figure 7: *The prediction depth can be the same, or very different for the same input when it occurs in the train and validation splits. Corners of this plot correspond to different forms of example difficulty.* (See Section 4 for discussion.)

Deep Learning Through the Lens of Example Difficulty (2021)

Kalibracja

- czy wartości zwracane przez model są dobrym przybliżeniem rzeczywistego prawdopodobieństwa?
- zapomniany problem z tradycyjnego ML (np. drzewa binarne są źle skalibrowane), który stał się ostatnio na powrót popularny

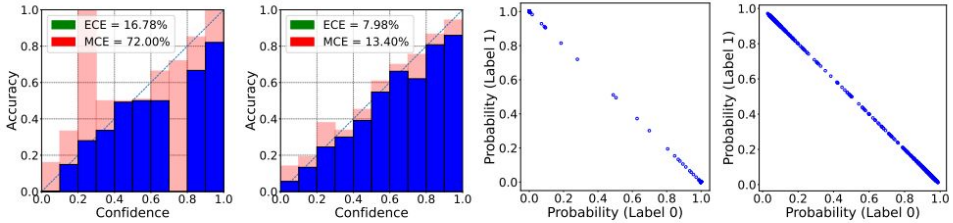
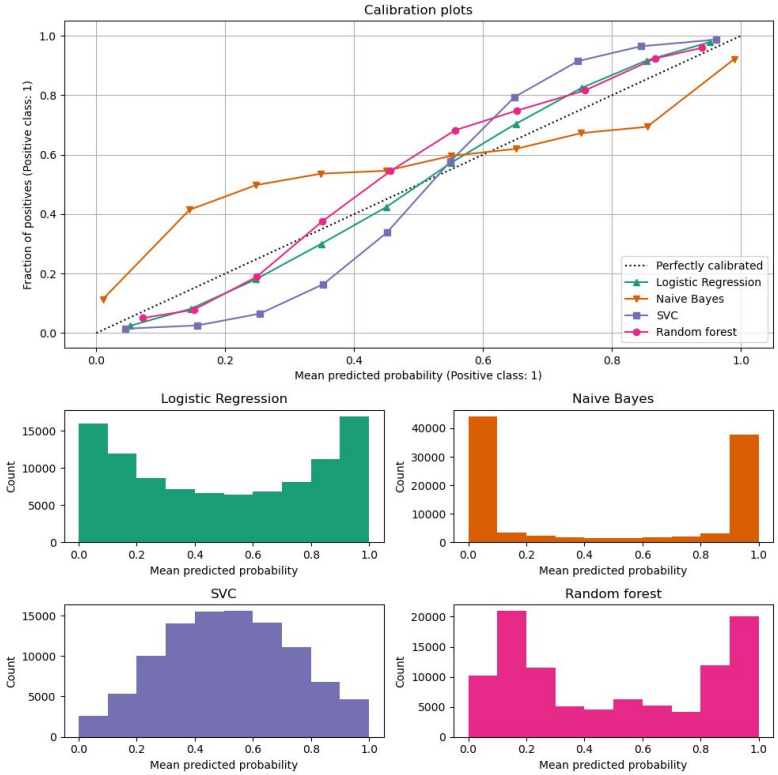


Figure 1: An illustration of model calibration. Uncalibrated model (left) that trained with standard cross-entropy loss and calibrated model (right) that trained with focal loss ($\gamma = 5$), have similar predictive performance on a binary classification task (accuracy is 83.8% and 83.4% respectively),



https://scikit-learn.org/1.5/modules/calibration.html
Calibration in Deep Learning: A Survey of the State-of-the-Art (2024)

Zła kalibracja: przykład

powinno być bliżej 1% dla obu graczy;
ta pozycja to praktycznie już remis

https://www.youtube.com/watch?v=mctyhQoK-tM

Losowe filtry konwolucyjne

- losowo zainicjalizowanych nietrenowanych filtrów konwolucyjnych można używać do *feature extraction*
- na tak stworzonych cechach można zastosować tradycyjny klasyfikator, np. Linear Classification z reg. L2, aby uzyskać prosty model
- podejście to pozwoliło osiągnąć wyniki state-of-the-art w Time Series Classification

```python
class RandomConvFilters(nn.Module):
    def __init__(self, input_is_grayscale, num_filters):
        super(RandomConvFilters, self).__init__()
        self.conv1 = nn.Conv2d(
            1 if input_is_grayscale else 3,
            num_filters,
            3
        )
        self.conv1.requires_grad = False

    def forward(self, x):
        x = self.conv1(x)
        return (x>0).float().mean(dim=[2,3]) # proportion of positive values


class LogisticRegression(nn.Module):
    def __init__(self, num_inputs, num_classes):
        super(LogisticRegression, self).__init__()
        self.fc1 = nn.Linear(num_inputs, num_classes)

    def forward(self, x):
        x = self.fc1(x)
        return x
```

MINIROCKET: A Very Fast (Almost) Deterministic Transform for Time Series Classification (2020)