Neprivalomos užduotys papildomam balui (II dalis, 0,5 b.)
Pateikti `emokymai.vu.lt` (užd. *Bonus*) iki gegužės 19 d. 23:59

Atsakydami į klausimus galite pasamprotauti, pateikti pavyzdžių, tačiau palankiai vertinami ir trumpi atsakymai. Pateikimo forma nėra svarbi – tinka tekstiniai failai, *notebooks*, užrašų, sprendimų nuotraukos, *PDFs*.

1. Kuo skiriasi stochastinis gradientinis nusileidimas (angl. *stochastic gradient descent*) nuo minirinkinių gradientinio nusileidimo (angl. *minibatch gradient descent*)? Kuriais atvejais naudotume vieną, kuriais kitą? (0,1 b.)

2. Kaip reiktų pakeisti stochastinio gradientinio nusileidimo algoritmą, jei kažkuri gradiento koordinatė būtų pastoviai didesnė nei kitos? (Pvz. gradiento $\nabla_\theta \mathcal{L} = \begin{bmatrix} 0.63 & -0.47 & 129.58 & 1.24 \end{bmatrix}$ trečioji koordinatė ženkliai didesnė nei kitos.) (0,1 b.)

3. Kaip reiktų pakeisti klasikinę tekstinę paiešką, kad ji galėtų atsakyti užklausą apytiksliai? Pateikite pavyzdį. Galite remtis *D2L* knyga (*Dive into Deep Learning Zhang et al. 2023. Chapter 11. Attention Mechanisms and Transformers*). (0,1 b.)

4. Spręsdami vaizdų lokalizavimo uždavinius dažnai modelio išvestyje gauname daugelį persidengiančių stačiakampių. Dažnai naudojame nemaksimalaus apjungimo (angl. *non-maximum suppression*) algoritmą, kad pašalintume didžiąją dalį stačiakampių. Šis algoritmas yra godus (angl. *greedy*)[1]. Ar gali atsitikti taip, kad su šiuo algoritmu pašalinsime naudingus stačiakampius? Kaip galėtume pakeisti šį algoritmą, kad ne taip griežtai išmestume stačiakampius, ir išgautume geresnius spėjimus? Galite remtis *Soft-NMS* šaltiniu (*Soft-NMS – Improving Object Detection With One Line of Code*, Bodla et al., 2017.). (0,1 b.)

5. Kartais *ReLU* aktyvacijos funkcija gali „pradingti“ (angl. *dead ReLU problem*) – t.y., su bet kokia įvestimi neurono išvestis lygi nuliui, o tai reiškia, kad jo išvestinė taip pat tampa lygi nuliui. Taip nutikus modelis lėčiau mokosi ir gali neišmokti kai kurių požymių. Kaip reiktų spręsti šią problemą? (0,1 b.)

[1] Godus algoritmas (angl. *greedy algorithm*) – algoritmas, kuris kiekviename etape priima lokaliai optimalius sprendimus. Kai kuriuose uždaviniuose lokaliai optimalūs sprendimai priveda prie globaliai optimalaus sprendimo (pvz. trumpiausio kelio grafe radimas naudojant *Dijkstra* algoritmą).