Deep Learningdagi asosiy tushunchalari o’zbekcha tariflari bilan:

1. **Deep Learning** esa ML ning kichik bir sohasi bo‘lib, neyron tarmoqlar yordamida katta hajmdagi ma’lumotlardan avtomatik ravishda andozalarni o‘rganadi. Bu usul qo'lda **feature engineering**ni talab qilmaydi, chunki u ma'lumotlarni bevosita o'zi o'rganadi.

Misol: Avtomatik boshqariladigan mashina yo‘ldagi obyektlarni taniydi.

Qisqacha aytganda, Deep Learning ancha kuchli, lekin ko‘proq ma’lumot va hisoblash quvvatini talab qiladi.

2. **Neural Network** – bu Deep Learning (DL) dagi sun’iy model bo‘lib, inson miyasidan ilhomlangan. U neyron qatlamlaridan iborat bo‘lib, ma’lumotlarni bosqichma-bosqich qayta ishlaydi.

Qanday ishlaydi?

🔷 **Input Layer** – Xom ma’lumotni qabul qiladi (masalan, rasm, matn yoki sonlar).

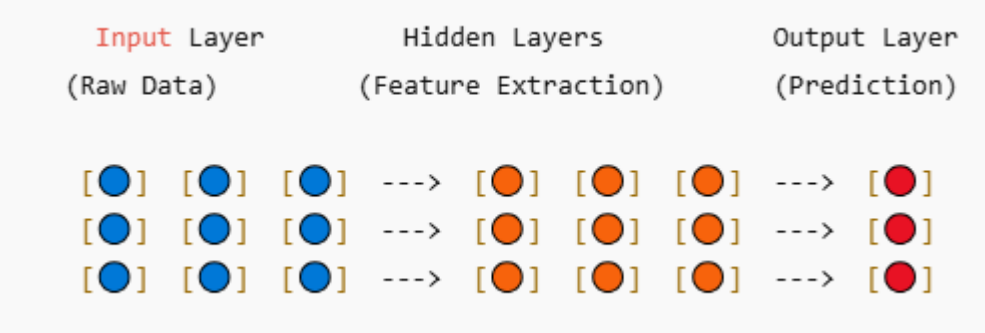
🔷 **Hidden Layers** – Andozalarni aniqlaydi (masalan, chekkalar, shakllar, obyektlar).

🔷 **Output Layer** – Yakuniy natijani chiqaradi (masalan, "Bu mushuk").

✅ Ma’lumotlardan avtomatik o‘rganadi

✅ Murakkab vazifalar uchun juda yaxshi (rasmni tanish, nutqni qayta ishlash va h.k.)

❌ Ko‘p ma’lumot va hisoblash quvvatini talab qiladi



3. **Deep Learning** tizimida weights va bias siljish asosiy o‘zgaruvchilardir, ular model tomonidan o‘rganiladi va natijalarni yaxshilash uchun moslanadi.

Weights, W: Bu har bir input ma’lumotning qanchalik muhimligini aniqlovchi koeffitsiyentdir. Neyronlar orasidagi bog‘lanishlarning har biri o‘ziga xos vaznga ega bo‘lib, model aynan shu qiymatlarni o‘zgartirib o‘rganadi.

Bias, b: Bu qo‘shimcha qiymat bo‘lib, hisob-kitob natijasiga qo‘shiladi. U modelga chiziqni yuqoriga yoki pastga siljitish imkonini beradi va natijani yanada moslashuvchan qiladi.

Oddiy Misol: Faraz qilaylik, uy narxini uning maydoni bo‘yicha bashorat qilmoqchimiz:

Vazn (W) – har kvadrat metr uchun qo‘shiladigan narx (masalan, 500 dollar/m^2).

Bias (b) – asosiy boshlang‘ich narx (masalan, 50 000 dollar).

Yakuniy hisoblash: Narx = (Maydon×W) +b

Deep Learning modelida aynan shu W va b qiymatlari training jarayonida backpropagation usuli yordamida moslashtiriladi va natijani yaxshilashga xizmat qiladi.

4. **Deep Learning** tizimida **loss function** modelning qanchalik yaxshi yoki yomon ishlayotganini o‘lchaydi. U bashorat qilingan natija (model chiqishi) bilan haqiqiy qiymat o‘rtasidagi farqni hisoblaydi. Model training jarayonida ushbu yo‘qotishni kamaytirishga harakat qiladi, shunda natijalar aniqroq bo‘ladi.

Oddiy Misol:

Faraz qilaylik, bolaga to‘pni savatga tashlashni o‘rgatyapsiz. Agar to‘p savatga tushmasa, siz to‘p savatdan qanchalik uzoq tushganini o‘lchaysiz—bu yo‘qotish (loss) degan tushuncha. Masofa qanchalik kichik bo‘lsa, tashlash shunchalik yaxshilanadi.

Asosiy Loss Functions:

1. MSE - Mean Squared Error – Regression masalalarida qo‘llanadi, bashorat va haqiqiy qiymat o‘rtasidagi kvadratik farqlarni hisoblaydi.
2. Cross-Entropy Loss – Klassifikatsiya masalalarida qo‘llanadi, model natijalari va haqiqiy belgilangan toifalar o‘rtasidagi farqni o‘lchaydi.

Model o‘zining **weights** va **bias** qiymatlarini o‘zgartirib, ushbu yo‘qotishlarni kamaytirishga harakat qiladi va natijalarni yanada aniqroq qiladi.

5. **Deep Learning** tizimida **tensors** asosiy ma’lumot tuzilmasi bo‘lib, ular ma’lumotlarni saqlash va qayta ishlash uchun ishlatiladi. Tensorlar massivlar (arrays) yoki matritsalar (matrices) ga o‘xshaydi, lekin ular bir nechta o‘lchamga ega bo‘lishi mumkin. TensorFlow va PyTorch kabi kutubxonalar aynan tenzorlar ustida tezkor matematik amallar bajaradi.

Tensorlarni sonlarni saqlovchi idishlar deb tasavvur qilaylik:

- **scalar** – bitta son, 0D tensor (masalan, 5).
- **vector** – sonlar ro‘yxati, 1D tensor (masalan, [3, 7, 2]).
- **matrix** – sonlar jadvali, 2D tensor (masalan, [[1, 2], [3, 4]]).

3D va undan yuqori o‘lchamli tensorlar – murakkab ma’lumotlar, masalan, tasvirlar va videolar uchun ishlatiladi.

Nima Uchun Tenzorlar Muhim?

- ✓ Ular GPU yordamida tezkor hisob-kitoblarni bajarishga yordam beradi.
- ✓ Ular modelni o‘qitish uchun ma’lumotlarni saqlaydi va qayta ishlaydi.
- ✓ Ular bilan o‘lchamni o‘zgartirish, matritsa amallari, masshtablash kabi operatsiyalarni oson bajarish mumkin.

Deep Learning modellari aynan shu tensorlarni qatlamlar (layers) orqali o‘zgartirib, ularning qiymatlarini moslab, natijalarni yaxshilaydi.

6. Deep Learning tizimida **activation function** neural networkdagi neyronning "faollashishi" yoki yo‘qligini aniqlaydi. Bu funksiya non-linearity qo‘shadi va modelga murakkab andozalarni o‘rganish imkonini beradi.

Neyron tarmog‘ini qaror qabul qiluvchi tizim deb tasavvur qilaylik:

- Agar aktivatsiya funksiyasisiz bo‘lsa, u faqat oddiy hisob-kitoblar (chiziqli operatsiyalar) bajaradi.
- Agar aktivatsiya funksiyasi ishlatilsa, u murakkab bog‘liqliklarni tushunib, tasvir yoki nutqni aniqlash kabi muammolarni hal qila oladi.

Eng Ko‘p Ishlatiladigan Aktivatsiya Funksiyalari:

1. ReLU (Rectified Linear Unit) – Manfiy qiymatlar uchun 0, musbat qiymatlar uchun esa o‘sha qiymatni qoldiradi. Oddiy va ko‘pchilik modelda yaxshi ishlaydi.

$$f(x)=\max(0, x)$$

2. Sigmoid – Kiritilgan qiymatni 0 va 1 oralig‘iga o‘tkazadi. Ehtimollik asosidagi chiqishlar uchun foydali.
3. Tanh (Hyperbolic Tangent) – Sigmoidga o‘xshash, lekin natijalarni -1 va 1 oralig‘ida qaytaradi. Agar ma’lumot nol atrofida markazlashgan bo‘lsa, yaxshi ishlaydi.

Nima Uchun Muhim?

- ✓ Modelga murakkab naqshlarni o‘rganishga yordam beradi.
- ✓ Neyron tarmoqlarga no- Linear muammolarni hal qilish imkonini beradi.
- ✓ Har xil muammolar uchun turli aktivatsiya funksiyalari ishlatiladi.

7. Deep Learning tizimida **gradient** – bu funksiya qanday o‘zgarishini o‘lchaydigan qiymat bo‘lib, u modelning **W** va **b** parametrlarini qanday yangilash kerakligini ko‘rsatadi. Gradient yordamida model xatoni kamaytirish uchun to‘g‘ri yo‘nalishda o‘zgaradi.

Gradientni tog‘dan tushish (pastga harakat qilish) deb tasavvur qiling:

- Agar gradient katta bo‘lsa (ya’ni, tog‘ tik bo‘lsa), tezroq harakat qilamiz.
- Agar gradient kichik bo‘lsa (ya’ni, tog‘ tekis bo‘lsa), sekinroq harakat qilamiz.
- Maqsad – eng past nuqtaga (xatolikning minimal darajasiga) yetish.

Deep Learning’da Qanday Ishlaydi?

1. Gradientni hisoblash – Gradient modelning vaznlarini qaysi yo‘nalishda o‘zgartirish kerakligini ko‘rsatadi.
2. Update the Weights – Gradientdan ma’lum bir qismini (o‘rganish tezligi – learning rate) olib, vaznlarni o‘zgartiramiz.
3. Takrorlash – Ushbu jarayon xatolik minimal bo‘lgunga qadar davom etadi.

Muhim Roli:

- ✓ **Gradient Descent** algoritmida modelni optimallashtirish uchun ishlatiladi.
- ✓ **Backpropagation** orqali tarmoqning aniqroq o‘rganishiga yordam beradi.
- ✓ Model xatolarini minimallashtirib, aniq natijalar olishga yordam beradi.

8. Deep Learning tizimida **batch size** – bu modelni training jarayonida bir vaqtning o‘zida ishlov beriladigan ma’lumotlar sonini bildiradi. Ma’lumotlar barcha bir vaqtning o‘zida ishlov berilmasdan, kichik qismlarga bo‘linadi va har bir qism alohida qayta ishlanadi.

Faraz qilaylik, siz yangi narsani o‘rganmoqdasiz, lekin hamma narsani bir vaqtning o‘zida o‘rganish o‘rniga, uni kichik qismlarga ajratib o‘rganasiz.

- Agar siz 1 bo‘limni o‘rganayotgan bo‘lsangiz, bu batch size = 1 ga o‘xshaydi (stoxastik gradient tushishi).
- Agar siz 10 bo‘limni o‘rganayotgan bo‘lsangiz, bu batch size = 10 ga o‘xshaydi.
- Batch size kattaroq bo‘lsa, model har bir qadamda ko‘proq ma’lumotni qayta ishlaydi, ammo ko‘proq xotira talab qiladi.

Nima Uchun Muhim?

- Kichik batch size: Model tezroq o‘rganadi, ammo natijalar noisy bo‘lishi mumkin va kamroq barqaror bo‘ladi.
- Katta batch size: Model barqarorroq ishlaydi, ammo sekinroq va ko‘proq xotira talab qiladi.
- Batch size tanlovi mavjud apparatga va ma’lumotlarning xususiyatiga qarab aniqlanadi.

Asosiy Nuqta:

- ✓ To‘g‘ri batch size tanlash modelni samarali o‘qitishda muhim ahamiyatga ega bo‘lib, tezlik va xotira foydalanishni muvozanatlashga yordam beradi.

9. Deep Learning tizimida **epoch** – bu butun o‘qitish ma’lumotlar to‘plamidan bir marta o‘tish jarayonini bildiradi. Har bir epoch davomida model barcha o‘qitish ma’lumotlarini qayta ishlaydi va xatolikni kamaytirish uchun vaznlar va siljishlarni yangilaydi.

Epochni kitobni boshidan oxirigacha o‘qish kabi tasavvur qilaylik:

- Bitta epochda siz butun kitobni (ma’lumotlar to‘plamini) o‘qiysiz.
- Bir nechta epochda kitobni bir necha marta o‘qib, har safar yaxshiroq tushunishga erishasiz (modelning yaxshilanishi).

Nima Uchun Muhim?

- ✓ Ko‘proq epochlar modelga ma’lumotlardan ko‘proq o‘rganish imkonini beradi.
- ✓ Juda kam epochlar modelning yetarlicha o‘rganmasligiga olib kelishi mumkin.
- ✓ Juda ko‘p epochlar overfitting (ortiqcha o‘rganish) ga olib kelishi mumkin, bunda model ma’lumotlarni yodlab qoladi va umumlashtira olmaydi.

Asosiy Nuqta:

To‘g‘ri epochlar soni modelning o‘rganish va umumlashtirish o‘rtasidagi muvozanatni saqlashga yordam beradi va yangi ma’lumotlarga yaxshi moslashishni ta’minlaydi.