Kapitola 5

Transfer Learning

V této kapitole se zaměříme na „transfer learning“, což je jedna z nejvýkonnějších technik v moderní umělé inteligenci. Tato metoda umožňuje využít znalosti, které se model naučil při řešení jedné úlohy, a aplikovat je na úlohu jinou, podobnou. Díky „transfer learning“ nemusíme trénovat složité modely „od nuly“, což šetří obrovské množství času, výpočetního výkonu i potřebných dat. Tato technika stojí za úspěchem moderních systémů pro rozpoznávání obrazu i pokročilých jazykových modelů.

Cíle kapitoly *(po přečtení této kapitoly budete schopni)*

C1.1. Vysvětlit, co je „transfer learning“ a proč je v praxi tak důležitý.
C1.2. Popsat rozdíl mezi trénováním „od nuly“ („from scratch“) a přenosem znalostí.
C1.3. Vysvětlit pojmy „pre-trained model“, „fine-tuning“ a „feature extraction“.
C1.4. Popsat proces „zmrazení“ („freezing“) vrstev a úpravu klasifikační hlavy.
C1.5. Uvést praktické příklady využití v počítačovém vidění, zpracování textu apod.

Klíčové myšlenky *(po přečtení této kapitoly byste si měli pamatovat)*

KM1.1. „Transfer learning“ využívá dříve naučené vzorce (například tvary v obrázcích) pro nové úkoly.
KM1.2. Modely se učí hierarchicky – první vrstvy vidí obecné rysy, poslední vrstvy konkrétní objekty.
KM1.3. „Pre-trained“ modely jsou modely již natrénované na obrovských datasetech.
KM1.4. „Fine-tuning“ je proces jemného doladění vah předtrénovaného modelu na specifických datech.
KM1.5. „Freezing“ („zmrazení“) zabraňuje změně vah v určitých vrstvách modelu během tréninku.
KM1.6. „Transfer learning“ dramaticky snižuje množství dat potřebných pro dosažení vysoké přesnosti.

Nové příkazy *(po přečtení kapitoly se seznámím s těmito novými příkazy)*

```
models.resnet18(), nn.Linear(), param.requires_grad, nn.CrossEntropyLoss(),
optim.Adam(),model.train(),model.eval(),torch.no_grad()
```

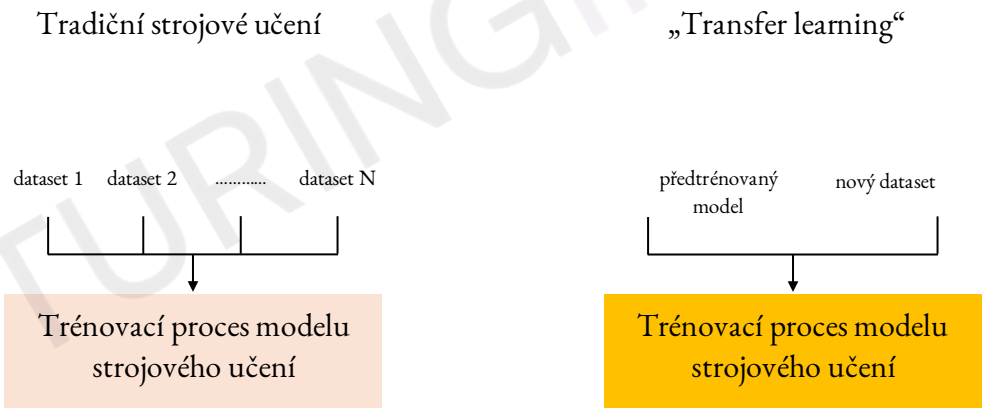

© Bee Learning s.r.o., IČO 10827161. Licence Turing Institut (turinginstitut.cz): materiál smí být použit pouze ve výuce Turing Institut. Komerční užití a použití k jakékoli jiné výuce je zakázáno.

1.1 Proč existuje „transfer learning“ (TL)?

Abychom pochopili přínos „transfer learning“ metod, musíme se podívat na to, jak funguje klasický přístup, který můžeme nazvat jako tzv. učení od nuly („training from scratch“). V tomto režimu začíná každý model jako „nepopsaný list“ a na svůj úkol se dívá, jako by nikdy předtím nic neviděl. Pokud bychom například vyvíjeli systém pro rozpoznávání dopravních značek v autonomním vozidle, musela by se klasická neuronová síť z tisíců fotografií nejprve pracně naučit základní statistické a geometrické vlastnosti obrazu – co je to čára, jak vypadá barva, co je to kruh a jak se liší objekt od pozadí. Teprve po tréninku na obrovském množství trénovacích dat a výpočetního času by byla tato síť schopna rozpoznat, že konkrétní kruh s číslem „50“ znamená omezení rychlosti. Tento přístup je však pro komplexní úlohy prakticky neudržitelný z několika klíčových důvodů:

1. nedostatek dat – málokterý specifický projekt má k dispozici miliony označených vzorků, které hluboké sítě vyžadují pro „pochopení“ světa,
2. výpočetní neefektivita – je nesmyslné učit auto znovu „vidět“ kruh nebo čáru pokaždé, když do systému přidáváte rozpoznávání nového typu značky.

„Transfer learning“ (schematicky lze vidět na obrázku níže) tento princip trénování od nuly zásadně narušuje. Místo abychom budovali celý model neustále znovu, využíváme model, který již prošel „školu“ na milionech jiných objektů (zvířatech, domech, lidech apod.) – předtrénovaný model, tedy model, který byl již dříve natrénován na velkém datasetu. Takový model již dávno rozumí tomu, jak vypadá hrana nebo barevný přechod. My pouze doplníme specifickou část modelu (typicky poslední vrstvy) a dotrénujeme ho, jak tyto již známe tvary interpretovat v kontextu silničního provozu – tedy jak správně poznat dopravní značení.



Obrázek : Schéma funkce přístupu „transfer learning“.

Poznámka k vysvětlujícím příkladům v tomto učebním textu:

V tomto učebním textu jsou vysvětlující příklady záměrně voleny převážně z oblasti zpracování obrazu, protože umožňují intuitivní pochopení principů. Je však důležité si uvědomit, že popisované přístupy nejsou omezeny pouze na obrazová data. Stejné principy se uplatňují i v dalších oblastech, například ve zpracování textu, zvuku nebo časových řad, kde se rovněž využívají předtrénované modely a jejich následné přizpůsobení konkrétním úlohám.

© Bee Learning s.r.o., IČO 10827161. Licence Turing Institut (turinginstitut.cz): materiál smí být použit pouze ve výuce Turing Institut. Komerční užití a použití k jakékoli jiné výuce je zakázáno.

1.2 Proč jsou znalosti přenositelné?

Abychom pochopili, proč je možné znalosti mezi modely přenášet, musíme se podívat na to, jak hluboké sítě zpracovávají vstupní data. Sítě se neučí data jako jeden celek, ale vytvářejí hierarchické reprezentace ve formě vrstev (učí se pochopit data vrstvu po vrstvě):

1. Základní vrstvy (nízkoúrovňové znalosti či reprezentace): první vrstvy sítě fungují jako detektory základních rysů. U obrázků rozpoznávají hrany a barvy, u textů vztahy mezi slovy apod. Tyto rysy jsou **univerzální**, hrana na fotce psa je z pohledu neuronové sítě reprezentována velmi podobně jako hrana na dopravní značce.
2. Střední vrstvy (středně-úrovňové znalosti či reprezentace): střední vrstvy skládají čáry do základních tvarů (kruhy, mřížky) nebo slova do frází. Tyto znalosti jsou stále velmi široce použitelné v mnoha aplikacích.
3. Hluboké vrstvy (vysokoúrovňové znalosti či reprezentace): teprve poslední vrstvy sítě (často označované jako „klasifikační hlava“) interpretují struktury z minulých vrstev na reálné konkrétní objekty – například „toto je značka omezující rychlost na 50 km/h“.

„Transfer learning“ tedy funguje proto, že nižší a střední vrstvy zachycují obecné rysy, které lze napříč úlohami předpokládat jako velmi podobné, či dokonce stejné. Při použití této techniky tedy „recyklujeme“ hotové základní a střední vrstvy předtrénovaných modelů a vyvíjíme už pouze specifické vrstvy – realizujeme tedy dotrénování těchto předtrénovaných modelů na specifické aplikace, kterými mohou být právě detekce omezení rychlosti, detekce chybovosti průmyslových součástek apod.

1.3 Proces TL – modifikace architektury, proces zmrazení a „fine-tuning“

Jakmile si vybereme vhodný předtrénovaný model (například ResNet, který je „expertem“ na tvary), musíme provést jeho technickou úpravu pro náš konkrétní úkol a jeho následný „fine-tuning“. Tento proces lze rozdělit do tří základních fází:

1. Modifikace výstupní vrstvy

   Původní předtrénovaný model má na svém konci určitý počet výstupních jednotek (neuronů), který odpovídá jeho původnímu úkolu – například 1000 neuronů pro 1000 druhů zvířat. Pro náš úkol detekce dopravních značek musíme tuto poslední vrstvu nahradit novou. Zásadním pravidlem je, že počet neuronů v této nové vrstvě musí přesně odpovídat počtu tříd, které chceme klasifikovat – například 10 typů značek = 10 neuronů.

2. Zmrazení vah („freezing“)

   Abychom při našem dotrénování modifikovaného modelu z bodu 1 nezničili užitečné znalosti uložené v jeho dřívějších vrstvách, musíme váhy v těchto vrstvách označit jako netrénovatelné („non-trainable“, během tréninku nebude docházet k jejich aktualizaci). Tím zajistíme, že se při výpočtu gradientu a následné aktualizaci vah budou měnit parametry pouze v naší nové výstupní vrstvě. Zmrazené vrstvy fungují jako neměnný filtr, který z našich dat extrahuje rysy, které se naučil už dříve.

3. Doladění předtrénovaného modelu neboli „fine-tuning“

© Bee Learning s.r.o., IČO 10827161. Licence Turing Institut (turinginstitut.cz): materiál smí být použit pouze ve výuce Turing Institut. Komerční užití a použití k jakékoli jiné výuce je zakázáno.

Po základním natrénování nové výstupní vrstvy můžeme přistoupit k doladění modelu, což se označuje jako tzv. „fine-tuning“. V této fázi postupně „rozmrazíme“ vybrané hlubší vrstvy neuronové sítě a dovolíme jim aktualizaci parametrů.

Riziko:
Při tomto procesu však narážíme na riziko poškození dříve naučených reprezentací. Pokud je rychlost učení (tzv. „learning rate“) příliš vysoká, chyby z nového tréninku (dotrénování) mohou způsobit příliš velké skoky v trénovatelných parametrech modelu (váhy a biasy). To vede k tomu, že model „zapomene“ univerzální vzorce (např. jak vypadá kruh, dochází k jevu známému jako „catastrophic forgetting“) a soustředí se pouze na šum v našich nových datech. „Fine-tuning“ se proto provádí s výrazně nižší rychlostí učení než původní trénink, často řádově o jeden až dva řády menší.

Výsledkem tohoto procesu je finální model, který je přizpůsoben našemu konkrétnímu úkolu a dokáže správně vyhodnocovat i data, která během tréninku neviděl. Model přitom nevyužívá pouze informace z našeho trénovacího datasetu, ale staví na znalostech, které se naučil již při předtrénování na původních rozsáhlých datech. Díky tomu jsme schopni dosáhnout velmi dobrých výsledků rychleji a s menším množstvím dat, než kdybychom model trénovali úplně od začátku.

1.4 Způsoby použití „transfer learning“ přístupu v praxi

V předchozí části jsme si ukázali, jak „transfer learning“ technicky funguje. V praxi však neexistuje pouze jeden správný postup použití. To, jakým způsobem předtrénovaný model použijeme, závisí především na množství dostupných dat pro náš konkrétní úkol a na tom, jak moc se tento úkol podobá tomu, na kterém byl předtrénovaný model původně trénován. V zásadě se dají rozlišit dva základní přístupy (lze vidět i na obrázku č. 2):

1. Použití předtrénovaného modelu jako extraktoru příznaků („feature extraction“)

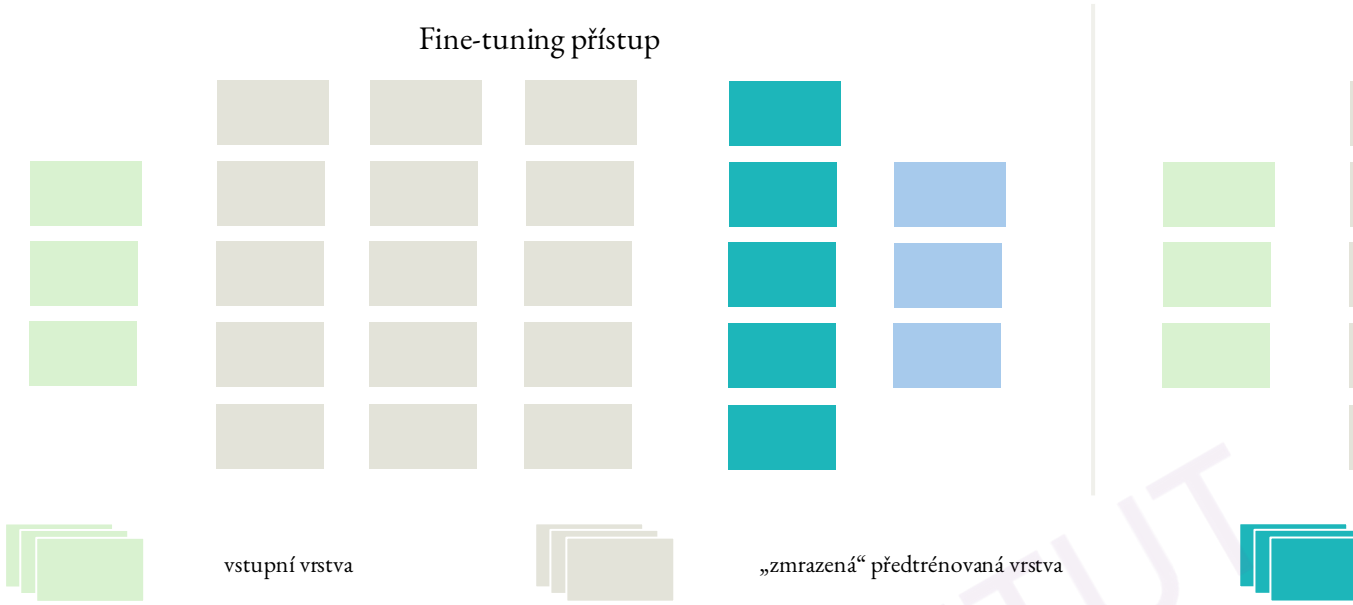
   V tomto přístupu využíváme předtrénovaný model pouze jako nástroj pro získání užitečných rysů z dat. Všechny původní vrstvy modelu ponecháme zmrazené (nedovolíme aktualizaci jejich trénovatelných parametrů) a trénujeme pouze novou výstupní vrstvu (nebo větší počet vrstev), kterou jsme přidali pro náš konkrétní úkol. Předtrénovaný model tedy funguje jako pevný filtr: například z obrázku nejprve extrahuje hrany, tvary a další obecné rysy, které se naučil již dříve, a nová vrstva se pouze učí, jak tyto rysy správně interpretovat. Tento přístup je vhodný zejména v případě, kdy máme k dispozici jen malé množství nových trénovacích dat pro náš konkrétní úkol. Díky tomu, že neměníme většinu modelu, je trénink stabilní a riziko přeučení („overfitting“) je relativně malé.

2. Doladění modelu („fine-tuning“)

   Druhým přístupem je tzv. „fine-tuning“, který jsme si již představili v předchozí části. V tomto případě kromě nové výstupní vrstvy postupně „odemykáme“ i některé hlubší vrstvy modelu a dovolujeme jim upravit své trénovatelné parametry podle našich nových dat. Model se tak nepřizpůsobuje pouze na úrovni finálního rozhodování (výstupní vrstva), ale částečně upravuje i samotné vnitřní reprezentace dat. Díky tomu může dosahovat lepších výsledků, zejména pokud se naše nová data liší od těch, na kterých byl předtrénovaný model původně trénován. Na druhou stranu je tento přístup náročnější na správné nastavení

© Bee Learning s.r.o., IČO 10827161. Licence Turing Institut (turinginstitut.cz): materiál smí být použit pouze ve výuce Turing Institut. Komerční užití a použití k jakékoli jiné výuce je zakázáno.

trénovacího procesu (například rychlosti učení) a při nevhodném použití může dojít ke zhoršení výkonu modelu.



Obrázek : Schematické rozdělení přístupů „transfer learning“ na „feature extraction“ a „fine-tuning“.

Nabízí se tedy otázka: jak zvolit vhodný přístup? Volba mezi těmito dvěma přístupy naštěstí není náhodná a řídí se několika jednoduchými pravidly:

- pokud máme málo nových dat pro náš úkol, je vhodnější použít pouze extrakci příznaků (případ č. 1),
- pokud jsou naše nová data velmi podobná původním datům, často stačí pouze extrakce příznaků (případ č. 1),
- pokud máme více nových dat, můžeme si dovolit „fine-tuning“ (případ č. 2),
- pokud se nová data výrazně liší, je vhodné model alespoň částečně doladit (případ č. 2).

V praxi se často postupuje takto: nejprve se model natrénuje pouze s novou výstupní vrstvou („feature extraction“, případ č. 1) a teprve poté, když je to potřeba, se přechází k jemnému doladění vybraných vrstev („fine-tuning“, případ č. 2).

1.5 Jak poznat, že „transfer learning“ bude fungovat

Ne vždy je „transfer learning“ tou nejlepší volbou. Abychom jej mohli efektivně využít, je důležité pochopit, kdy dává smysl, a kdy naopak nemusí fungovat dobře. Hlavním faktorem je podobnost mezi daty, na kterých byl model původně trénován, a daty, která chceme použít pro náš nový úkol.

Podobnost dat jako klíčový faktor

Model se během předtrénování naučí určité vzorce – například hrany, tvary nebo struktury v obraze. Tyto vzorce ale dávají smysl pouze tehdy, pokud se podobné struktury vyskytují i v našich nových datech. Čím více se tedy naše nová data podobají těm původním, tím větší část znalostí dokáže model efektivně využít.

Příklady z praxe

© Bee Learning s.r.o., IČO 10827161. Licence Turing Institut (turinginstitut.cz): materiál smí být použit pouze ve výuce Turing Institut. Komerční užití a použití k jakékoli jiné výuce je zakázáno.

- Velmi podobná data – model trénovaný na fotografiích zvířat použijeme pro rozpoznávání jiných zvířat – „transfer learning“ bude fungovat **velmi dobře**.
- Středně podobná data – model trénovaný na běžných fotografiích použijeme pro rozpoznávání průmyslových součástek – „transfer learning“ často funguje, ale je vhodné použít „fine-tuning“.
- Velmi odlišná data – model trénovaný na běžných fotografiích použijeme pro klasifikaci na lékařských snímcích (například rentgen) – přenos znalostí pomocí přístupu „transfer learning“ může být velmi omezený a celkový přístup nemusí fungovat.

Přístup efektivního dotrénování již předtrénovaného modelu bude tedy fungovat nejlépe tehdy, když mezi původními a novými daty existuje smysluplná podobnost. Pokud jsou však nová data příliš odlišná od těch původních, může model využívat nevhodné reprezentace a jeho výkon se nemusí zlepšit, ale naopak zhoršit. Tento jev se označuje jako „negative transfer“. V takovém případě je často vhodné model více upravit, nebo zvážit jeho trénování od začátku.

1.6 Praktické cvičení s řešením

V tomto cvičení si ukážeme, jak využít „transfer learning“ v praxi. Použijeme předtrénovaný model a přizpůsobíme jej jednoduché klasifikační úloze. Cílem je pochopit celý proces:

- načtení předtrénovaného modelu,
- úprava jeho struktury,
- zmrazení vrstev a
- dotrénování na nových datech.

Zadání: Použijte předtrénovaný model ResNet18 a upravte jej tak, aby dokázal klasifikovat obrázky do 10 tříd (dataset CIFAR-10).

Poznámka ke kódu:
V reálných úlohách se kromě trénovací a testovací množiny často používá i validační množina, podle které ladíme hyperparametry a sledujeme, zda model nezačíná přeučovat. Pro jednoduchost však tato validační množina nebyla v ukázkovém kódu zahrnuta.

Krok 1 – Import knihoven

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
```

Krok 2 – Příprava dat

```
transform = transforms.Compose([
    transforms.Resize((224, 224)),  # ResNet očekává větší obrázky
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5),
                         (0.5, 0.5, 0.5))
])

train_dataset = datasets.CIFAR10(root='./data', train=True,
                                 download=True, transform=transform)
```


© Bee Learning s.r.o., IČO 10827161. Licence Turing Institut (turinginstitut.cz): materiál smí být použit pouze ve výuce Turing Institut. Komerční užití a použití k jakékoli jiné výuce je zakázáno.

```
test_dataset = datasets.CIFAR10(root='./data', train=False,
                                download=True, transform=transform)

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

Krok 3 – Načtení předtrénovaného modelu

```
model = models.resnet18(pretrained=True)
```

Krok 4 – Úprava výstupní vrstvy

```
model.fc = nn.Linear(model.fc.in_features, 10)
```

Krok 5 – Zmrazení vrstev („feature extraction“ přístup)

```
for param in model.parameters():
    param.requires_grad = False

for param in model.fc.parameters():
    param.requires_grad = True
```

Krok 6 – Definice ztrátové funkce a optimalizačního algoritmu

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.fc.parameters(), lr=0.001)
```

Krok 7 – Dotrénování předtrénovaného modelu

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

for epoch in range(5):
    model.train()
    running_loss = 0.0

    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    print(f"Epoch {epoch+1}, Loss: {running_loss:.4f}")
```

Krok 8 – Vyhodnocení dotrénovaného modelu

```
model.eval()
correct = 0
total = 0

with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
```


© Bee Learning s.r.o., IČO 10827161. Licence Turing Institut (turinginstitut.cz): materiál smí být použit pouze ve výuce Turing Institut. Komerční užití a použití k jakékoli jiné výuce je zakázáno.

```
            outputs = model(images)

            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

print(f"Přesnost modelu: {100 * correct / total:.2f} %")
```

Rozšíření pro náročnější studenty:

Zkuste model dále vylepšit:

- „rozmrazte“ poslední konvoluční blok a proveďte „fine-tuning“,
- porovnejte výsledky s modelem trénovaným od nuly,
- změňte „learning rate“ a sledujte vliv na výkon modelu.

TURINGINSTITUT

© Bee Learning s.r.o., IČO 10827161. Licence Turing Institut (turinginstitut.cz): materiál smí být použit pouze ve výuce Turing Institut. Komerční užití a použití k jakékoli jiné výuce je zakázáno.