# OCR Prevodilac

Soft kompjuting – Vukašin Janković, SW/29-2015
Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka, Novi Sad

## Uvod

Računarska inteligencija i kompjuterska vizija u obradi slike se primenjuju danas u mnogim aspektima života i zastupljeni su u skoro svim pametnim uređajima. Ideja ovog projekta je primena istih u optičkom prepoznavanju karaktera (OCR) i ekstrakciji teksta sa slike kao i prosleđivanje prepoznatog teksta prevodiocu koji za uneti početni i odredišni jezik prevodi dati tekst. Pun potencijal ovog projekta se može videti u budućoj mobilnoj aplikaciji koja bi direktno pristupala kameri, slikala tekst i vršila instant prevod.

Aplikacija podržava cifre, sva slova engleske abecede kao i srpska slova š,đ,č,ć,ž.

## Arhitektura i tehnologije

Projekat je rađen na *Python 3.6 + Anaconda 5.2.0* platformi i u programskom okruženju *Jupyter Notebook.*

Korišćene biblioteke za OCR:

- *OpenCV*
- *Keras*
- *Scikit*

Za prevod teksta je korišćena biblioteka *googletrans* koja sadrži ugrađen Google Translate API.
**Napomena:** S obzirom da se za prevod teksta samo pozivaju ugrađene metode ove biblioteke i da je akcenat projekta na samom prepoznavanju teksta, ova komponenta će biti isključena iz dalje analize i evaluacije.

## Struktura

Ceo projekat se sastoji iz 3 osnovne faze: ***Treniranje neuronske mreže, Ekstrakcija teksta sa slike i Prevođenje***.

Proces treniranja traje ~45min i korišćeno je 5 različitih fontova:

- **ABCDabcd1234šđčćž**
- ***ABCDabcd1234šđčćž***
- ***ABCDabcd1234šđčćž***
- **ABCDabcd1234šđčćž**
- ABCDabcd1234šđčćž

Sam OCR proces se sastoji iz sledećih faza:

1. *Učitavanje slike*
2. *Predprocesiranje (binarizacija, uklanjanje šuma, rotacija)*
3. *Ekstrakcija kontura i sortiranje*
4. *Prepoznavanje*

## Učitavanje i predprocesiranje slike

Obradićemo sledeći primer:


Svaku sliku je prvo neophodno konvertovati u format pogodan za izdvajanje kontura. To ćemo postići binarizacijom slike i uklanjanjem šuma. Naša aplikacija nudi sledeće metode pronalaženja *threshold-a* za binarizaciju:

- Otsu (default)
- Adaptive mean
- Adaptive gaussian
- Scikit local

Šum se uklanja kombinacijom **dilacije** i **erozije** a takođe je moguće blurovati sliku korišćenjem *median* i *gaussian* metode.
Nakon obrade, naša slika dobija novi izgled:


Primetićemo da je tekst ispisan pod uglom ili nekom perspektivnom, te je nephodno ispraviti dati tekst kako bi prepoznavanje karaktera bilo uspešnije. Korišćenjem određenih metoda detektujemo ceo tekst na slici:


Nakon obrade, dobijamo relativno ravan tekst:


## Ekstrakcija kontura i sortiranje

Obrađenu sliku šaljemo na detekciju kontura i spajanje odvojenih delova na slova ukoliko postoje (i,š,ž,č,ć) i dobijamo rezultat:


Osim samih regiona, čuvamo i distance između istih kako bi mogli da spojimo prepoznate karaktere u slova i redove. Izdvojene regione sortiramo po odgovarajućem redosledu i dobijamo podatke spremne za slanje istreniranoj neuronskoj mreži na prepoznavanje

## Prepoznavanje

Nakon obrade, naša mreža je prepoznala sledeći tekst:

```
Optžcai character Recosnžtžon
```

Empirijski gledano, naša aplikacija je za dati primer omašila u 5 slova što daje tačnost od **81%**

## Zaključak

Za jednostavne slike, ravnomernog osvetljenja i krupnog teksta, naša aplikacija prepoznaje slova sa visokim procentom preciznosti. Učitana slika prolazi kroz sve faze obrade i pripreme za prepoznavanje karaktera, čime je cilj projekta uspešno ispunjen.

Za kompleksnije slike, preciznost značajno opada. S obzirom da je svaka slika sebi specifična po određenim karakteristikama (osvetljenje, kontrast, orijentacija...), kvalitet rešenja u ovakvim slučajevima se može unaprediti treniranjem mreže nad većim skupom podataka kao i dodatnim "eksperimentisanjem" sa postojećim tehnikama *thresholding-a*, uklanjanja šuma, izdvajanja i sortiranja kontura dok se ne dođe do najoptimalnijeg rešenja. Takođe, postojeće metode se mogu unaprediti dodatnim proverama i obradama radi poboljšanja rezultata na kompleksnijim slikama.

Poređeni su rezultati naše aplikacije sa postojećim OCR algoritmima poput *Google Tesseract-a* i za krupan, ravan i jasan tekst na slici je *Tesseract* u skoro svim slučajevima davao bolji rezultat, dok je za neke slike sa mutnijom pozadinom i "iskrivljenim" tekstom naša aplikacija prevagnula.